5
25
43
55
59
68
75

1 कम्प्यूटर प्रणाली का हार्डवेयर संगठन
Hardware Organisation of Computer System

खण्ड 'अ' : अतिलघु उत्तरीय प्रश्न

प्रश्न 1. एसोसिएटिव मेमोरी का प्रत्येक वर्ड कितने बिट्स का होता है?

उत्तर एसोसिएटिव मेमोरी का प्रत्येक वर्ड N बिट्स का होता है।

प्रश्न 2. RISC निर्देश क्या है?

उत्तर RISC की फुल फॉर्म Reduced Instruction Set Computer है। इसकी आर्किटेक्चर में कुछ ऐसी प्रक्रिया होती है जिसमें कम्प्यूटर के सेट ऑफ इन्स्ट्रक्शन को सुलझाकर उनके द्वारा execution समय को कम कर दिया जाता है। इससे execution में भी कम समय लगता है। यह एक single निर्देश तन्त्र रखता है और इसमें फिक्सड length निर्देश होते हैं। इसमें कम ही एड्रैसिंग मोड होते हैं। इसमें पाइप लाइनिंग सुविधाजनक होती है।

खण्ड 'ब' : लघु एवं दीर्घ उत्तरीय प्रश्न

प्रश्न 1. सामान्य रजिस्टर ऑर्गेनाइजेशन (General Register Organisation) को चित्र सहित समझाइए।

उत्तर **सामान्य रजिस्टर ऑर्गेनाइजेशन (General Register Organisation)** प्रोग्रामिंग (Programming) करते समय प्वॉन्इटर्स (Pointers), काउन्टर्स (Counters), रिटर्न एड्रैसेज (Return Addresses), अस्थायी परिणामों (Temporary Results) और आंशिक गुणनफलों (Partial Products), इत्यादि को सुरक्षित करने के लिए मेमोरी लोकेशन्स (Memory Locations) की आवश्यकता होती है।

उपरोक्त कार्यों को करने के लिए मेमोरी (Memory) का प्रयोग करने पर काफी समय लग सकता है, क्योंकि मेमोरी-एक्सेस (Memory-Access) कम्प्यूटर में सबसे ज्यादा समय लेता है। अत: इन मानों को प्रोसेसर रजिस्टर में ही सुरक्षित करना बेहतर उपाय है। जब सीपीयू (CPU) में अनेक रजिस्टर्स (Registers) का प्रयोग किया जाता है तो उन्हें कॉमन बस (Common Bus) के द्वारा जोड़ना ही सर्वोत्तम है।

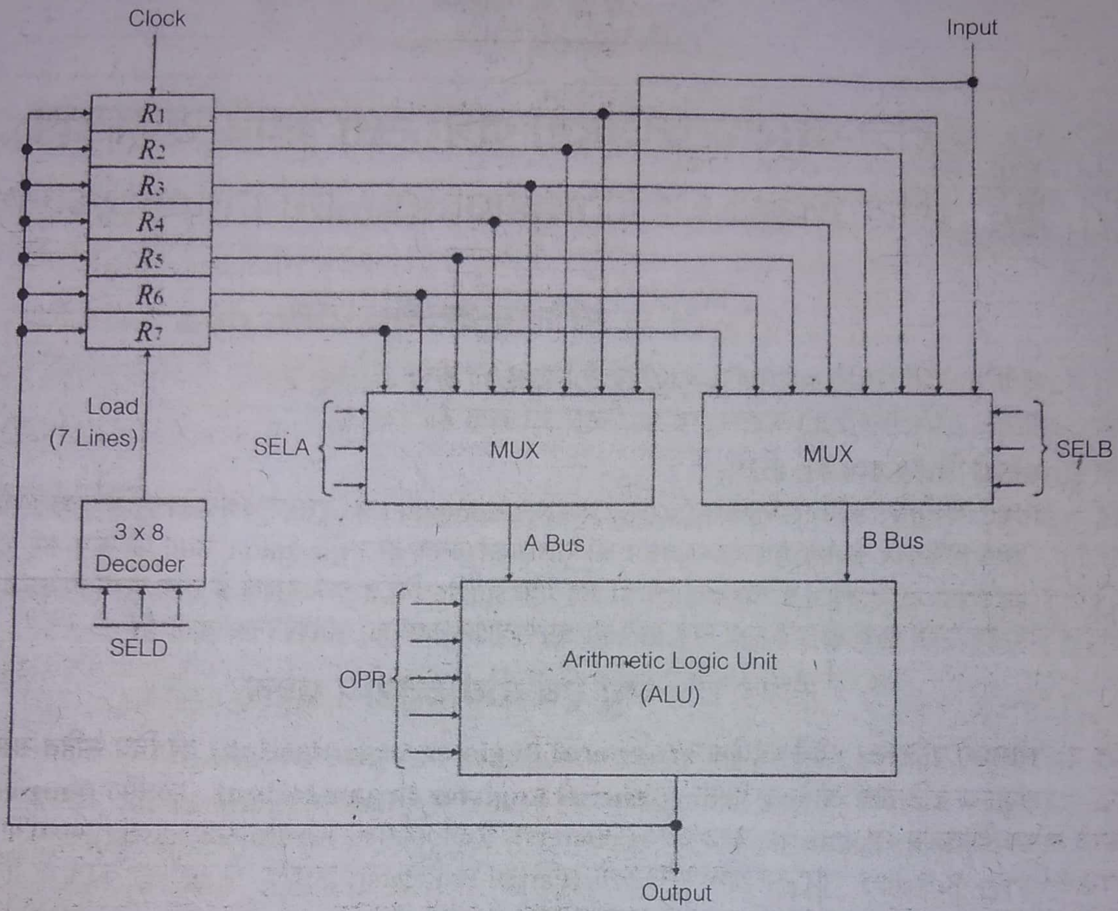
रजिस्टर को एक-दूसरे से संचार की आवश्यकता माइक्रोऑपरेशन (Microoperation) के एग्जिक्यूशन (Execution) के दौरान और डेटा के लेन-देन के लिए पड़ती है। अत: एक इस प्रकार की यूनिट की आवश्यकता है जो प्रोसेसर के सभी अर्थमेटिक (Arithmetic), लॉजिक (Logic) और शिफ्ट माइक्रोऑपरेशन (Shift Microoperation) को कर सके।

रजिस्टर (Register) के मध्य संचार (Communication) के लिए हमें कॉमन बस (Common Bus) की आवश्यकता होती है। सीपीयू (CPU) के 7 रजिस्टर्स के लिए बस (Bus) की व्यवस्था को चित्र 1.1 में दर्शाया गया है।

प्रत्येक रजिस्टर का आउटपुट दो मल्टीप्लेक्सर्स (Multiplexers-MUX) में दो बसेज (Buses) A और B को बनाने के लिए प्रवाहित किया जाता है। प्रत्येक मल्टीप्लेक्सर (Multiplexer) में सेलेक्शन लाइन्स (Selection Lines) किसी रजिस्टर (Register) अथवा विशेष बस (Bus) के लिए इनपुट (Input) को चुनती है। A और B बसेज (Buses) अर्थमेटिक लॉजिक यूनिट (Arithmetic Logic Unit) के लिए इनपुट (Input) प्रवाहित करती है।

ALU में चुना गया ऑपरेशन (Operation) यह निर्धारित करता है कि प्राप्त इनपुट पर कौन-सा अर्थमेटिक (Arithmetic) अथवा लॉजिक (Logic) माइक्रोऑपरेशन (Microoperation) किया जाना है, तत्पश्चात् आउटपुट (Output) को प्रवाहित कर दिया जाता है।

Scanned with CamScanner



चित्र 1.1

प्रश्न 2. रजिस्टर ऑर्गेनाइजेशन से आप क्या समझते हैं? इसके प्रकारों का वर्णन कीजिए।

उत्तर **रजिस्टर ऑर्गेनाइजेशन** Register Organisation एक सीपीयू (CPU) में, मेन मेमोरी (Main Memory) और कैशे (Cache) से ऊपर रजिस्टर (Register) का उपयोग किया जाता है। रजिस्टर (Register) का प्रयोग CPU सीधे ही कर सकता है। CPU में निम्नलिखित दो प्रकार के रजिस्टर्स (Registers) होते हैं—

1. **प्रोग्रामर-विजिबल रजिस्टर** Programmer-Visible Register एक प्रोग्रामर विजिबल रजिस्टर (Programmer-Visible Register), वह रजिस्टर (Register) है, जो सीपीयू (CPU) द्वारा एग्जिक्यूट होने वाले मशीनी भाषा के प्रोग्राम द्वारा रेफर (Refer) किया जा सकता है। दूसरे शब्दों में वह रजिस्टर (Register) जिसके मान को मशीनी भाषा द्वारा परिवर्तित अथवा प्रयोग किया जा सके, प्रोग्रामर-विजिबल रजिस्टर (Programmer-Visible Register) कहलाता है।

   इन्हें निम्नलिखित समूहों में वर्गीकृत किया जा सकता है—

   (i) **सामान्य उद्देश्य रजिस्टर** General Purpose Register प्रोग्रामर द्वारा रजिस्टर को विभिन्न कार्य दिए जा सकते हैं। सामान्य उद्देश्य रजिस्टर (General Purpose Register) में किसी भी ऑपकोड (Opecode) के ऑपरेन्ड (Operand) को सुरक्षित किया जा सकता है। परन्तु कुछ रजिस्टर्स (Registers) केवल फ्लोटिंग-प्वॉइन्ट संख्याओं (Floating Point Numbers) के लिए भी प्रयोग किये जा सकते हैं।

   (ii) **डेटा रजिस्टर** Data Register इनका उपयोग केवल डेटा को सुरक्षित करने के लिए किया जाता है। ये किसी भी आपरेन्ड (Operand) के एड्रैस की गणना का हिस्सा नहीं बन सकते हैं।

Scanned with CamScanner

(iii) **एड्रैस रजिस्टर** Address Register एड्रैस रजिस्टर का उपयोग एड्रैस (Address) को सुरक्षित करने के लिए किया जाता है। इन्डैक्स रजिस्टर (Index Register) और स्टैक प्वॉइन्टर (Stack Pointer) भी इसी प्रकार के रजिस्टर हैं।

2. **कण्ट्रोल एण्ड स्टेटस रजिस्टर** Control and Status Register सीपीयू (CPU) के ऑपरेशन्स का नियन्त्रण करने के लिए सीपीयू (CPU) में विभिन्न प्रकार के रजिस्टर का उपयोग किया जाता है। अधिकतर कम्प्यूटर्स में इनमें से अधिकतर प्रोग्रामर को विजिबल (Visibles) नहीं होते हैं और कुछ को कन्ट्रोल अथवा ऑपरेटिंग सिस्टम कोड (Control or Operating System Code) में मशीनी भाषा द्वारा योग किया जा सकता है। एक निर्देश (Instruction) को एग्जिक्यूट (Execute) करने में निम्नलिखित चार मुख्य रजिस्टर्स की आवश्यकता होती है—

(i) **प्रोग्राम काउन्टर** Program Counter इस रजिस्टर में एग्जिक्यूट होने वाले अगले निर्देश (Instruction) का एड्रैस (Address) सुरक्षित होता है। दूसरे शब्दों में इस रजिस्टर में फैच (Fetch) किये जाने वाले रजिस्टर का एड्रैस होता है।

(ii) **इन्सट्रक्शन रजिस्टर** Instruction Register इस रजिस्टर में एग्जिक्यूट किये जाने वाले निर्देश (Instruction) को सुरक्षित किया जाता है। दूसरे शब्दों में इस रजिस्टर में फैच (Fetch) किये गये निर्देश को सुरक्षित किया जाता हे।

(iii) **मेमोरी एड्रैस रजिस्टर** Memory Address Register इस रजिस्टर में रीड (Read) किये गये अथवा राइट (Write) किये जाने वाले डेटा का एड्रैस रखा जाता है।

(iv) **प्रोग्राम स्टेट्स वर्ड** Program Status Word यह एक रजिस्टर अथवा अधिक रजिस्टर्स का समूह है, जो स्टेट्स (Status) की सूचना को धारण करता है। प्रोग्राम स्टेटस वर्ड (Program Status Word-PSW) में स्टेट्स की सूचना, कॉमन फील्ड्स (Common Fields) अथवा फ्लैग्स (Flags) के अतिरिक्त कन्डीशन कोड (Condition Code) भी होते हैं। इनमें निम्नलिखित सम्मिलित होते हैं—

Sign (S) : अन्तिम अर्थमेटिक ऑपरेशन के परिणाम के चिह्न को सुरक्षित रखता है।

Zero (Z) : जब परिणाम 0 होता है तो यह सेट अथवा 1 हो जाता है।

Carry (C) : परिणाम के अनुसार यदि कैरी (Carry) योग में अथवा बौरो (Borrow) घटा में प्राप्त होती है, तो इसे सेट (Set) कर दिया जाता है।

Parity (P) : यदि परिणाम में 1 की संख्या सम होती है तो इसे सेट अथवा 1 कर दिया जाता है और यदि परिणाम में 1 की संख्या विषम होती है, तो इसे रीसेट अथवा 0 कर दिया जाता है।

प्रश्न 3. स्टैक ऑर्गेनाइजेशन (Stack Organization) पर टिप्पणी लिखिए।

उत्तर **स्टैक ऑर्गेनाइजेशन** Stack Organization अधिकांश कम्प्यूटर्स (Computers) के सीपीयू (CPU) में एक अत्यन्त महत्त्वपूर्ण गुण को सम्मिलित किया जाता है, जिसे स्टैक (Stack) अथवा लास्ट इन फर्स्ट आउट (Last In First Out-LIFO) लिस्ट (List) कहा जाता है। स्टैक (Stack) डेटा स्टोरेज डिवाइस (Data Storage Device) है, जो डेटा को इस प्रकार संचित करता है, कि जो तत्त्व सबसे अन्त में सुरक्षित होगा उसे पहले बाहर निकाला जायेगा। इसे समझने के लिये हम थालियों के ढेर से तुलना करते हैं। जहाँ हम सबसे पहले उस थाली को निकाल सकते हैं जो सबसे अन्त में रखी गयी हो।

डिजिटल कम्प्यूटर में स्टैक (Stack), एक ऐसी मेमोरी यूनिट (Memory Unit) है, जिसका एड्रैस (Address) गिना जा सकता है। वह रजिस्टर, जो स्टैक (Stack) का एड्रैस (Address) धारण (Hold) करने के लिये प्रयोग किया जाता है, उसे स्टैक प्वॉइन्टर (Stack Pointer-*SP*) कहा जाता है। यह सदैव स्टैक (Stack) में सबसे ऊपर (Top) स्टोर किये गये डेटा को प्वॉइन्ट (Point) करता है। एक स्टैक (Stack) में डेटा को डाला भी जा सकता है और डेटा को निकाला भी जा सकता है।

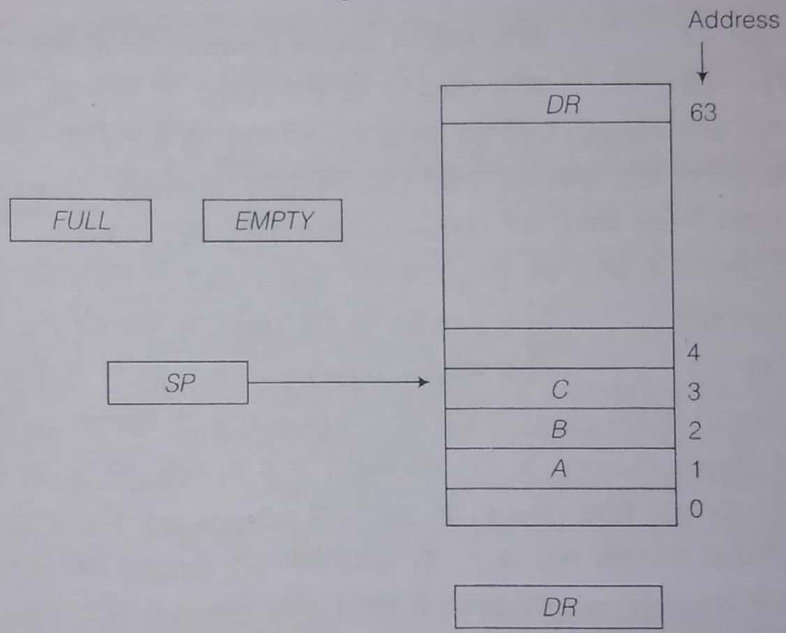
एक स्टैक (Stack) पर किये जा सकने वाले दो कार्य हैं— डेटा डालना अथवा डेटा इन्सर्शन (Data Insertion) और डेटा निकालना अथवा डेटा डिलिशन (Data Deletion)। एक स्टैक (Stack) में डेटा डालने के कार्य को पुश (Push)

Scanned with CamScanner

कहा जाता है, क्योंकि इसमें स्टैक (Stack) के टॉप (Top) पर डेटा को पुश (Push) किया जाता है। एक स्टैक (Stack) से डेटा को निकालने के कार्य को पॉप (Pop) कहा जाता है, क्योंकि स्टैक (Stack) के टॉप (Top) से डेटा को मिटाया अर्थात् पॉप-अप (Pop-Up) किया जाता है। इन कार्यों के दौरान स्टैक प्वॉइन्ट रजिस्टर (Stack Point Register) के मान को क्रमशः बढ़ाया और घटाया जाता है।

प्रश्न 4. रजिस्टर स्टैक (Register Stack) को चित्र सहित समझाइए।

उत्तर **रजिस्टर स्टैक** Register Stack स्टैक (Stack) को एक बड़ी मेमोरी (Memory) के एक भाग में भी बनाया जा सकता है अथवा उसे एक सीमित संख्या के मेमोरी वर्ड्स (Memory Words) अथवा रजिस्टर्स (Registers) के समूह द्वारा भी व्यवस्थित (Organize) किया जा सकता है। चित्र 1.2 में एक 64-वर्ड (Word) रजिस्टर स्टैक (Register Stack) के ब्लॉक डायग्राम (Block Diagram) को दर्शाया गया है—



चित्र **1.2** Block Diagram of A 64-bit Stack

स्टैक प्वॉइन्टर रजिस्टर (Stack Pointer Register) *SP* में दो बायनरी (Binary) संख्या सुरक्षित होंगी, जिसका मान स्टैक (Stack) में सबसे ऊपर (Top) सुरक्षित मान के एड्रैस के समान होता है अथवा दूसरे शब्दों में स्टैक प्वॉइन्ट रजिस्टर (Stack Point Register) में स्टैक (Stack) के टॉप (Top) का एड्रैस (Address) होता है चित्र 1.2 में तीन तत्त्व *A*, *B* और *C* सुरक्षित हैं। यहाँ स्टैक (Stack) के टॉप (Top) पर *C* सुरक्षित हैं। जिससे *SP* में *C* का एड्रैस (Address) सुरक्षित है, अतः *SP* का कन्टेन्ट 3 है। जब हमें स्टैक (Stack) से किसी तत्त्व को मिटाना होता है, तो पॉप (Pop) फंक्शन किया जाता है, स्टैक (Stack) के टॉप (Top) पर सुरक्षित तत्त्व को रीड (Read) कर लिया जाता है और फिर टॉप (Top) को घटा दिया जाता है। अतः हमारे उदाहरण में ऊपर सुरक्षित *C* को रीड (Read) किया जायेगा और जब घटाने के पश्चात् टॉप (Top) का एड्रैस (Address) 2 में हो जायेगा और अब स्टैक के टॉप (Top) पर *B* होगा। जब हमें स्टैक (Stack) में किसी तत्त्व को डालना होता है, तो पुश (Push) फंक्शन किया जाता है, जिसमें पहले स्टैक (Stack) के टॉप (Top) को बढ़ाया जाता है और फिर नये टॉप (Top) पर नये तत्त्व को राइट (Write) करके सुरक्षित कर दिया जाता है। जब हम स्टैक (Stack) में से किसी तत्त्व को मिटाते हैं, तो वह मिटता नहीं है, सिर्फ वहाँ से लिंक (Link) हट जाता है और जब दुबारा लिंक (Link) वहाँ आता है तो वहाँ नया तत्त्व सुरक्षित किया जा सकता है।

एक 64-वर्ड (word) के स्टैक (Stack) में, स्टैक प्वॉइन्टर (Stack Pointer) में सिर्फ 6 बिट्स (Bits) हो सकती हैं क्योंकि $2^6 = 64$। चूँकि स्टैक प्वॉइन्टर (Stack Pointer) केवल 6 बिट्स (Bits) का प्रयोग कर सकता है, इसलिए इसमें अधिकतम सुरक्षित एड्रैस (Address) का मान 63 (63 = 111111) हो सकता है। जब स्टैक (Stack) पूरा भरा

Scanned with CamScanner

होता है अर्थात् टॉप (Top) = 63, तब यदि हम टॉप के मान को बढ़ाते हैं तो, उसका मान 0 हो जायेगा क्योंकि 111111 + 1 = 1000000, और *SP* सिर्फ बायीं 6 बिट्स (Bits) को ही सुरक्षित कर सकता है। इसी प्रकार जब 000000 से 1 को घटाते हैं, तो उत्तर 111111 आयेगा। अत: दो रजिस्टर FULL और EMPTY प्रयोग किये जाते हैं। ये दोनों ही रजिस्टर 1 बिट के होते हैं। फुल (FULL) का मान तब 1 होगा, जब स्टैक (Stack) पूरा भरा होगा और एम्प्टी (EMPTY) का मान तब 1 होगा, जब स्टैक (Stack) पूरा खाली होता है। DR अर्थात् डेटा रजिस्टर (Data Register) एक ऐसा रजिस्टर है, जो स्टैक से रीड (Read) हो चुके अथवा स्टैक (Stack) में राइट (Write) होने वाले डाटा को सुरक्षित रखता है।

प्रश्न 5. पुश फंक्शन (Push Function) एवं पॉप फंक्शन (Pop Function) के बारे में समझाइए।

उत्तर **पुश फंक्शन** Push Function प्रारम्भ में *SP* में 0, EMPTY में 1 और FULL में 0 सुरक्षित होता है, जिसका तात्पर्य है कि *SP* एड्रैस (Address) 0 को दर्शा रहा है, स्टैक (Stack) खाली है और स्टैक (Stack) पूरा भरा नहीं है। जब भी हमें स्टैक (Stack) में कोई तत्त्व डालना होता है, तो सबसे पहले हम फुल (Full) का मान जाँचते हैं, यदि FULL = 0 है, अर्थात् स्टैक भरी नहीं है, तो स्टैक (Stack) में पुश ऑपरेशन (Push Operation) के द्वारा एक नया तत्त्व स्टैक (Stack) में डाला जा सकता है।

पुश ऑपरेशन (Push Operation) को कार्यान्वित करने के लिए माइक्रोऑपरेशन्स (Microoperations) की निम्नलिखित श्रेणी (Sequence) का प्रयोग किया जाता है—

$SP \leftarrow SP + 1$	स्टैक प्वॉइन्टर (Stack Pointer) के मान को बढ़ायें।
$M[SP] \leftarrow DR$	स्टैक (Stack) के टॉप (Top) पर तत्त्व को राइट (Write) करें।
If ($SP = 0$) then (FULL $\leftarrow$ 1)	जाँचें कि क्या स्टैक (Stack) भर गया है और यदि भर गया है तो FULL में 1 करें।
$EMPTY \leftarrow 0$	स्टैक (Stack) खाली नहीं है, चिह्नित करें।

यहाँ सबसे पहले स्टैक प्वॉइन्टर (Stack Pointer) *SP* के मान को 1 से बढ़ाया जाता है, जिससे स्टैक (Stack) वांछित नये टॉप (Top) को चिह्नित करें। तत्पश्चात् *DR* अर्थात् डेटा रजिस्टर में सुरक्षित तत्त्व को *SP* में सुरक्षित एड्रैस (Address) अर्थात् *M*[*SP*] पर राइट (Write) करें। यहाँ *M*[*SP*], *SP* में सुरक्षित एड्रैस (Address) द्वारा बताये गये मेमोरी वर्ड (Memory Word) को प्रदर्शित करता है। स्टैक (Stack) में पहला तत्त्व, एड्रैस 1 पर और अन्तिम तत्त्व 0 पर सुरक्षित होता है, अत: यदि *SP* 0 पर पहुँच गया है, तो इसका तात्पर्य है कि स्टैक (Stack) भर गया है, इसलिए रजिस्टर FULL में 1 सुरक्षित करें। अब चूँकि स्टैक (Stack) में एक तत्त्व लिखा गया है, अत: स्टैक (Stack) खाली नहीं है, इसलिये रजिस्टर EMPTY में 0 सुरक्षित करें।

पॉप फंक्शन Pop Function एक तत्त्व को स्टैक (Stack) में से तभी मिटाया जा सकता है, जब स्टैक (Stack) खाली नहीं है अर्थात् स्टैक (Stack) में से किसी तत्त्व को मिटाने के लिए EMPTY = 0 होना चाहिए। पॉप ऑपरेशन (Pop Operation) को कार्यान्वित करने के लिए माइक्रोऑपरेशन्स (Microoperations) की निम्नलिखित श्रेणी (Sequence) का प्रयोग किया जाता है—

$DR \leftarrow M[SP]$	स्टैक (Stack) के टॉप (Top) से डेटा को रीड (Read) करके *DR* में सुरक्षित करें।
$SP \leftarrow SP - 1$	स्टैक प्वॉइन्टर (Stack Pointer) के मान को घटायें।
If ($SP = 0$) then (EMPTY – 1)	जाँचें कि क्या स्टैक (Stack) खाली हो गया है, यदि हाँ, EMPTY $\leftarrow$ 1।
Full $\leftarrow$ 0	चिह्नित करें कि स्टैक (Stack) भरा नहीं हो।

यहाँ सर्वप्रथम *SP* द्वारा चिह्नित एड्रैस (Address) से डेटा को *DR* में सुरक्षित किया जाता है। तत्पश्चात् *SP* के मान में से 1 को घटा दिया जाता है। अब जाँचते हैं कि क्या *SP* का मान 0 हो गया है, यदि ऐसा होता है, तो इसका तात्पर्य है कि स्टैक (Stack) खाली हो गया है और इसलिए EMPTY = 1 कर दिया जाता है। यह स्थिति तभी उत्पन्न होगी जब स्टैक (Stack) के 1 एड्रैस से डेटा को रीड (Read) कर लिया जायेगा। अब चूँकि स्टैक (Stack) में से एक तत्त्व निकाल लिया गया है, अत: स्टैक (Stack) भरा नहीं है, इसलिए रजिस्टर FULL में 0 सुरक्षित करें।

Scanned with CamScanner

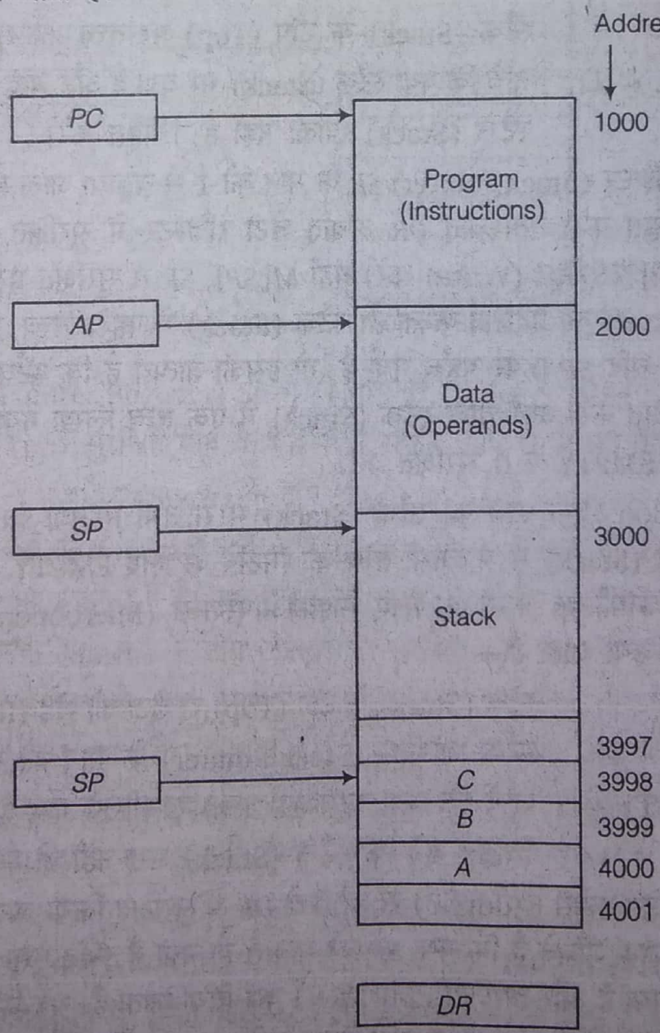
नोट करें कि FULL = 1 होने पर पुश ऑपरेशन (Push Operation) और EMPTY = 1 होने पर पॉप ऑपरेशन (Pop Operation) करने से सदैव गलत उत्तर आयेगा।

प्रश्न 6. मेमोरी स्टैक (Memory Stack) का वर्णन कीजिए।

उत्तर **मेमोरी स्टैक** (Memory Stack) स्टैक (Stack) एक स्वतन्त्र इकाई (Stand Alone Unit) की भाँति भी हो सकता है, साथ ही स्टैक (Stack) को CPU से जुड़ी हुयी रेन्डम ऐक्सेस मेमोरी (Random Access Memory) में भी कार्यान्वित किया जा सकता है। मेमोरी (Memory) के एक भाग को स्टैक (Stack) के रूप में और प्रोसेसर रजिस्टर (Processor Register) को स्टैक प्वॉइन्टर (Stack Pointer) के रूप में प्रयोग करके स्टैक (Stack) को CPU में कार्यान्वित किया जा सकता है। दिए गए चित्र 1.3 में कम्प्यूटर मेमोरी (Computer Memory) को प्रोग्राम, डेटा और स्टैक सेगमेन्ट्स (Stack Segments) के साथ प्रदर्शित किया गया है।

यहाँ कम्प्यूटर मेमोरी (Computer Memory) को तीन भागों—प्रोग्राम (Program), डेटा (Data) और स्टैक (Stack) में विभाजित किया गया है। प्रोग्राम काउन्टर (Program Counter) *PC* प्रोग्राम के अगले निर्देश को चिह्नित करता है। ऐरे रजिस्टर (Array Register) *AR* डेटा के ऐरे (Array) को चिह्नित करता है। स्टैक प्वॉइन्टर (Stack Pointer) *SP* स्टैक (Stack) के टॉप (Top) को चिह्नित करता है।

तीनों रजिस्टर एक कामन एड्रैस बस (Common Address Bus) से जुड़े होते हैं और कोई भी एक मेमोरी (Memory) के लिए एड्रैस (Address) उपलब्ध करा सकता है। फैच (Fetch) के दौरान किसी निर्देश को रीड (Read) करने के लिए *PC* का उपयोग किया जाता है।



चित्र 1.3 Computer Memory with Program, Data and Stack Segments

Scanned with CamScanner

एग्जिक्यूशन (Execution) के दौरान ऑपरेन्ड (Operand) को रीड (Read) करने के लिए *AR* का उपयोग जाता है। *SP* का प्रयोग स्टैक (Stack) में तत्त्वों को पुश (Push) अथवा पॉप (Pop) करने के लिए किया जाता है। जैसा कि चित्र 1.3 में दर्शाया गया है कि *SP* का प्रारम्भिक मान 4001 है और स्टैक (Stack) में ऊपर बढ़ते हुए (Address) घट रहा है, अत: स्टैक (Stack) पहला तत्त्व एड्रैस (Address) 4000 पर, दूसरा 3999 पर, और 3998 पर सुरक्षित है और स्टैक (Stack) में प्रयोग किया जा सकने वाला अन्तिम एड्रैस (Address) 3000 है। स्टैक (Stack) की सीमायें जाँचने के लिए कोई प्रबन्ध नहीं है।

हम मान कर चलते हैं कि स्टैक (Stack) से रीड (Read) अथवा राइट (Write) होने वाला डेटा *DR* में सुरक्षि अब एक नये तत्त्व को मेमोरी (Memory) में डालने के लिए पुश (Push) फंक्शन (Function) के निर्देश निम्न हैं—

$SP \leftarrow SP - 1$ स्टैक प्वॉइन्टर (Stack Pointer) *SP* के मान को घटायें।

$M[SP] \leftarrow DR$ स्टैक (Stack) के टॉप (Top) पर तत्त्व को राइट (Write) करें।

चूँकि यहाँ स्टैक (Stack) बढ़ते हुए *SP* के मान को घटा रहा है इसलिए पहले *SP* के मान को घटायेंगे, ताकि ये शब्द को चिह्नित करें। तत्पश्चात् मेमोरी राइट ऑपरेशन (Memory Write Operation) के द्वारा *DR* में सुरक्षि को *SP* द्वारा चिह्नित मेमोरी वर्ड (Memory Word) पर राइट (Write) करें। स्टैक (Stack) के टॉप (Top) तत्त्व को मिटाने के लिए पॉप (Pop) फंक्शन (Function) के निर्देश निम्नलिखित हैं—

$DR \leftarrow M[SP]$ स्टैक (stack) के टॉप (top) से डेटा को रीड (read) कर *DR* में सुरक्षित करें।

$SP \leftarrow SP + 1$ स्टैक प्वॉइन्टर (stack pointer) *SP* के काम को बढ़ायें।

स्टैक (Stack) के टॉप (Top) से डेटा को रीड (Read) करके रजिस्टर *DR* में सुरक्षित किया जाता है। तत्पश्चात प्वॉइन्टर (Stack Pointer) के मान को बढ़ाया जाता है, ताकि वह अगले तत्त्व को चिह्नित करें।

अधिकतर कम्प्यूटर स्टैक (Stack) के ओवरफ्लो (Overflow) और अन्डरफ्लो (Underflow) को जाँचने के कोई हार्डवेयर उपलब्ध नहीं कराते हैं। ओवरफ्लो (Overflow) का तात्पर्य है, भरा हुआ स्टैक (Stack) अन्डरफ्लो (Underflow) का तात्पर्य है खाली स्टैक (Stack)। स्टैक (Stack) की सीमाओं को जाँचने के लि प्रोसेसर रजिस्टर्स (Processor Registers) एक ऊपरी सीमा के लिए, इस उदाहरण में 3000, तथा दूसरा निचली के लिए, इस उदाहरण में 4001, का प्रयोग किया जा सकता है।

पुश (Push) अथवा पॉप (Pop) ऑपरेशन (operations) करते समय दो माइक्रोऑपरेशन (Microopera पहला *SP* के द्वारा मेमोरी (Memory) का प्रयोग करना तथा दूसरा *SP* के मान में परिवर्तन करना, की आवश्यक है। कौन-सा ऑपरेशन कहाँ और किस प्रकार किया जाना है, यह स्टैक ऑर्गेनाइजेशन (Stack Organizatio निर्भर करता है। उदाहरण के लिए, *SP* के मान में परिवर्तन के लिए उसके मान को बढ़ाना अथवा घटाना है, यह इ पर निर्भर करता है कि स्टैक (Stack) को किस क्रम में बनाया गया है।

प्रारम्भ में स्टैक प्वॉइन्टर (Stack Pointer) में आरम्भिक मान को डाला जाता है। यह आरम्भिक मान (Memory) में स्टैक के निचले एड्रैस (Address) में 4000 ही होना चाहिए। इसके पश्चात् माइक्रोऑ (Microoperation) के अनुसार ही *SP* के मान को बढ़ाया व घटाया जा सकता है।

प्रश्न 7. इन्स्ट्रक्शन फॉरमेट्स (Instruction Formats) को विस्तारपूर्वक समझाइए।

उत्तर **इन्स्ट्रक्शन फॉरमेट्स** Instruction formats कम्प्यूटर के फिजिकल और लॉजिकल स्ट्रक्चर (Physical Logical Structure of Computer) की व्याख्या सामान्यत: सिस्टम (System) के साथ दिये गये रेफरेन्स मै (Reference Manual) में कर दी जाती है। इस प्रकार के मैनुअल (Manual) सीपीयू (CPU) की आन्तरिक (Internal Construction) को वर्णित करते हैं। इनमें प्रोसेसर रजिस्टर्स (Processor Registers) और लॉजिकल क्षमता (Logical Capability) भी परिभाषित होती है। इनमें हार्डवेयर द्वारा कार्यान्वित (Hard Implemented) सभी निर्देश, उनके बायनरी कोड फॉरमेट (Binary Code Format) और प्रत्येक निर्देश की

Scanned with CamScanner

परिभाषा की सूची दी होती है। एक कम्प्यूटर में सामान्यत: अनेक प्रकार के इन्सट्रक्शन कोड फॉरमेट्स (Instruction Code Formats) का प्रयोग किया जाता है। सीपीयू (CPU) में प्रत्येक इन्सट्रक्शन कोड (Instruction Code) को इन्टरप्रेट (Interpret) करने और निर्देश को पूरा करने के लिए आवश्यक कन्ट्रोल फंक्शन (Control Functions) को उपलब्ध कराने का कार्य कन्ट्रोल यूनिट (Control Unit) का होता है।

एक निर्देश (Instruction) को फॉरमेट (Format) को सामान्यत: एक आयताकार बॉक्स (Rectangular Box) द्वारा चित्रित किया जाता है, जो निर्देश (Instruction) की बिट्स (Bits) को उसी प्रकार चिह्नित करता है, जिस प्रकार वे मेमोरी वर्ड्स (Memory Words) अथवा कन्ट्रोल रजिस्टर (Control Register) में प्रदर्शित होंगे। निर्देश (Instruction) की बिट्स (Bits) को समूहों (Groups) में विभाजित कर दिया जाता है, जिन्हें फील्ड (Field) कहा जाता है। इन्सट्रक्शन फॉरमेट (Instruction Format) में पाये जाने वाले मुख्य फील्ड्स (Fields) निम्नानुसार हैं—

1. **ऑपरेशन कोड फिल्ड** Operation Code Field यह फील्ड किये जाने वाले कार्य अथवा ऑपरेशन (Operation) को परिभाषित करती है।
2. **एड्रैस फील्ड** Address Field यह फील्ड मेमोरी एड्रैस (Memory Address) अथवा प्रोसेसर रजिस्टर (Processor Register) को प्रदर्शित करती है।
3. **मोड फील्ड** Mode Field यह फील्ड ऑपरेन्ड (Operand) अथवा इफेक्टिव एड्रैस (Effective Address) को निश्चित करने के तरीके को परिभाषित करती है।

अन्य विशेष फील्ड्स (Fields) का उपयोग विशेष स्थितियों में किया जाता है।

एक निर्देश (Instruction) की ऑपरेशन कोड फील्ड (Operation Code Field), बिट्स (Bits) का एक समूह है, जो प्रोसेसर (Processor) के विभिन्न ऑपरेशन्स (Operations) को परिभाषित करती है; उदाहरण के लिए, योग (Add), घटा (Subtract), कॉम्प्लीमेन्ट (Complement) और शिफ्ट (Shift)। एक निर्देश की मोड फील्ड (Mode Field) में दी गयीं बिट्स (Bits), दिये गये एड्रैस (Address) से ऑपरेन्ड (Operand) को चुनने के विभिन्न विकल्पों को परिभाषित करती हैं।

कम्प्यूटर निर्देशों (Computer Instructions) द्वारा उल्लेखित किये गये ऑपरेशन्स (Operations) को मेमोरी (Memory) अथवा प्रोसेसर रजिस्टर्स (Processor Register) पर एग्जिक्यूट (Execute) किया जाता है। मेमोरी (Memory) वाले ऑपरेन्ड्स (Operands) को मेमोरी एड्रैस (Memory Address) और प्रोसेसर रजिस्टर्स (Processor Register) वाले ऑपरेन्ड्स (Operands) को रजिस्टर एड्रैस (Register Address) द्वारा उल्लेखित (Specify) किया जाता है। रजिस्टर एड्रैस (Register Address) R बिट्स (Bits) की एक बायनरी संख्या (Binary Number) होती है, जो CPU के 2^R रजिस्टर्स (Registers) में से किसी एक रजिस्टर (Register) को परिभाषित करती है। अत: 16 प्रोसेसर रजिस्टर (Processor Register) वाले सीपीयू (CPU) में $R0$ से $R15$ तक रजिस्टर (Register) को प्रदर्शित करने के लिए, 4 बिट्स (Bits) के रजिस्टर एड्रैस (Register Address) की आवश्यकता होगी। उदाहरण के लिए, बायनरी संख्या 0111 रजिस्टर $R7$ को चिह्नित करेगी। एक कम्प्यूटर में विभिन्न लेन्थ (Length) के निर्देश हो सकते हैं, जिनमें एड्रैस (Address) की संख्या परिवर्तित हो सकती है। एक कम्प्यूटर में एक निर्देश (Instruction) में एड्रैस फील्ड्स (Address Fields) की संख्या उसके रजिस्टर्स (Registers) के आन्तरिक ऑर्गेनाइजेशन (Internal Organisation) पर निर्भर करती है। अधिकांश कम्प्यूटर्स निम्नलिखित तीनों में से किसी एक CPU ऑर्गेनाइजेशन (Organisation) का प्रयोग करते हैं—

1. सिंगल एक्यूमुलेटर ऑर्गेनाइजेशन (Single Accumilator Organisation)
2. जनरल रजिस्टर ऑर्गेनाइजेशन (General Register Organisation)
3. स्टैक ऑर्गेनाइजेशन (Stack Organisation)

एक्यूमुलेटर-टाइप ऑर्गेनाइजेशन (Accumulator Type Organisation) में सभी ऑपरेशन्स (Operations) एक्यूमुलेटर (Accumulator) की सहायता ली जाती है। इस प्रकार के इन्सट्रक्शन फॉरमेट (Instruction Format) में

Scanned with CamScanner

केवल एक एड्रैस फील्ड (Address Field) होती है। उदाहरण के लिए, असेम्बली लैंग्वेज निर्देश (Assembly Language Instruction) द्वारा परिभाषित एक निर्देश (Instruction) जो अर्थमेटिक एडीशन (Arithmetic Addition) है, निम्नानुसार है—

```
ADD X
```

जहाँ X ऑपरेन्ड (Operand) का एड्रैस (Address) है। यहाँ ADD निर्देश $AC \leftarrow AC + M|X|$ ऑपरेशन को करेगा। AC एक्यूमुलेटर रजिस्टर (Accumulator Register) है, और $M|X|$, X एड्रैस (Address) पर सुरक्षित मेमोरी वर्ड (Memory Word) को प्रदर्शित करता है।

जनरल रजिस्टर टाइप ऑर्गेनाइजेशन (General Register Type Organisation) के इन्सट्रक्शन फॉरमेट (Instruction Format) में कम्प्यूटर को तीन रजिस्टर एड्रैस फील्ड्स (Register Address Fields) की आवश्यकता होती है। इस प्रकार के अर्थमेटिक एडीशन (Arithmetic Addition) को असेम्बली लैंग्वेज (Assembly Language) में निम्नानुसार प्रदर्शित किया जा सकता है—

```
ADD R1, R2, R3
```

यह निर्देश (Instruction) $R1 \leftarrow R2 + R3$ ऑपरेशन (Operation) को प्रदर्शित करता है। यदि गंतव्य रजिस्टर (Destination Register), स्रोत रजिस्टर्स (Source Registers) में से ही कोई एक हो, तो एड्रैस फील्ड्स (Address Fields) की संख्या को तीन से घटाकर दो किया जा सकता है। अत: निर्देश (Instruction)

```
Add R1, R2
```

ऑपरेशन (Operation) $R1 \leftarrow R1 + R2$ को प्रदर्शित करता है। इस निर्देश (Instruction) में केवल $R1$ और $R2$ के रजिस्टर एड्रैस (Register Address) को ही उल्लेखित (Specify) किया गया है।

मल्टीपल प्रोसेसर रजिस्टर (Multiple Processor Register) वाले कम्प्यूटर मूव निर्देश (Move Instruction) के नीमोनिक (Mnemonic) MOV के साथ प्रयोग करके ट्रांसफर निर्देश (Transfer Instruction) को प्रदर्शित करते हैं। अत: निर्देश (Instruction)

```
MOV R1, R2
```

ऑपरेशन ट्रान्सफर (Transfer) $R1 \leftarrow R2$ (अथवा कम्प्यूटर विशेष पर निर्भर करने से $R2 \leftarrow R1$) को प्रदर्शित कर रहा है। ट्रान्सफर टाइप इन्सट्रक्शन (Transfer Type Instruction) में स्रोत (Source) और गंतव्य (Destination) को परिभाषित करने के लिए दो एड्रैस फील्ड्स (Address Fields) की आवश्यकता होती है।

जनरल रजिस्टर टाइप (General Register Type) कम्प्यूटर के इन्सट्रक्शन फॉरमेट (Instruction Format) में दो अथवा तीन एड्रैस फील्ड्स (Address Fields) का प्रयोग किया जाता है। इसकी प्रत्येक एड्रैस फील्ड (Address Field) प्रोसेसर रजिस्टर (Processor Register) अथवा मेमोरी वर्ड (Memory Word) को प्रदर्शित कर सकती है। निम्नलिखित निर्देश (Instruction)

```
ADD R1, X
```

ऑपरेशन (Operation) $R1 \leftarrow R1 + M[X]$ को प्रदर्शित कर रहा है। इसमें दो एड्रैस फील्ड्स (Address Fields) हैं, जिसमें से एक प्रोसेसर रजिस्टर $R1$ को और दूसरा मेमोरी एड्रैस (Memory Address) X को प्रदर्शित कर रहा है।

स्टैक ऑर्गेनाइजेशन (Stack Organisation) वाले कम्प्यूटर्स में पुश (Push) और पॉप (Pop) निर्देशों (Instructions) का प्रयोग किया जाता है, जिन्हें एक एड्रैस फील्ड (address field) की आवश्यकता होती है। अत: निर्देश (Instruction)

```
Push X
```

X एड्रैस (Address) पर दिये गये वर्ड (Word) को स्टैक (Stack) के टॉप (Top) पर पुश (Push) करता है। स्टैक प्वॉइन्ट (Stack Pointer) स्वत: ही अपडेट (Update) हो जायेगा। स्टैक-ऑर्गेनाइज्ड कम्प्यूटर्स (Stack Organised Computers) के ऑपरेशन टाइप निर्देश (Operation Type Instruction) में किसी एड्रैस (Address) की आवश्यकता नहीं पड़ती है। ऐसा इसलिए होता है, क्योंकि स्टैक ऑर्गेनाइजेशन (Stack Organisation) में कोई भी

Scanned with CamScanner

ऑपरेशन (Operation) स्टैक (Stack) के दो टॉप मोस्ट (Top Most) आइटम्स (Items) पर किया जाता है। निर्देश (Instruction)

ADD

स्टैक (Stack) कम्प्यूटर्स में बिना किसी एड्रैस फील्ड (Address Field) का ऑपरेशन कोड (Operation Code) है। इस ऑपरेशन (Operation) के द्वारा स्टैक (Stack) की दो टॉप मोस्ट (top Most) संख्याओं को पॉप (Pop) करके जोड़ा जाता है, और योगफल को स्टैक (Stack) के टॉप (Top) पर भेज दिया जाता है। चूँकि सभी ऑपरेन्ड्स (Operands) स्टैक में होते हैं, इसलिए यहाँ ऑपरेन्ड्स (Operands) को बताने के लिए किसी एड्रैस फील्ड (Address Field) की आवश्यकता नहीं होती है।

चर्चा की गयी तीनों ऑर्गेनाइजेशन्स (Organisations) में से, अधिकतर कम्प्यूटर्स (Computers), किसी एक का प्रयोग करते हैं। कुछ कम्प्यूटर्स एक से अधिक ऑर्गेनाइजेशन स्ट्रक्चर (Organisation Structure) के गुणों (Features) को कम्बाइन (Combine) करके प्रयोग करते हैं। उदाहरण के लिए Intel 8080 में सात रजिस्टर होते हैं, जिनमें से एक एक्यूमुलेटर रजिस्टर (Accumulator Register) होता है। प्रोसेसर (Processor) में कुछ गुण (Features) जनरल रजिस्टर टाइप ऑर्गेनाइजेशन (General Register Type Organisation) के और कुछ गुण (Features) एक्यूमुलेटर टाइप (Accumulator Type) के होते हैं। Intel 8080 प्रोसेसर (Processor) में स्टैक प्वॉइन्टर (Stack Pointer) और मेमोरी स्टैक (Memory Stack) से पुश (Push) और पॉप (Pop) करने के लिये निर्देश (Instructions) भी होते हैं। फिर भी इस प्रोसेसर में स्टैक ऑर्गेनाइज्ड सीपीयू (Stack Organised CPU) वाले जीरो एड्रैस टाइप इन्स्ट्रक्शन (Zero-Address Type Instruction) नहीं होते हैं।

कम्प्यूटर प्रोग्राम में एड्रैसेज (Addresses) की संख्या का प्रभाव देखने के लिए हम निम्नलिखित अर्थमेटिक स्टेटमेन्ट (Arithmetic Statement) को हल करेंगे—

$$X = (A + B) * (C + D)$$

इस समीकरण को हम शून्य (Zero), एक (One), दो (Two) और तीन (Three) एड्रैस (Address) की सहायता से हल करेंगे। चारों अर्थमेटिक ऑपरेशन (Arithmetic Operation) को प्रदर्शित करने के लिए हम ADD, SUB, MUL और DIV चिह्नों (Symbols) का ट्रांसफर-टाइप ऑपरेशन (Transfer Type Operation) को प्रदर्शित करने के लिए MOV का; मेमोरी (Memory Operation) और AC रजिस्टर से ट्रान्सफर (Transfer) करने के लिए LOAD एवं STORE का प्रयोग करेंगे। हम यह मान कर चल रहे हैं कि चार ऑपरेन्ड्स (Operands), A, B, C और D हैं और उत्तर को मेमोरी एड्रैस (Memory Address) X पर सुरक्षित किया जायेगा।

प्रश्न 8. तीन-एड्रैस निर्देश, दो-एड्रैस निर्देश, एक-एड्रैस निर्देश तथा शून्य-एड्रैस निर्देश को समझाइए।

उत्तर **तीन-एड्रैस वाले निर्देश** Three-Address Instruction थ्री-एड्रैस इन्सट्रक्शन फॉरमेट (Three-Address Instruction Format) को प्रयोग करने वाले कम्प्यूटर में प्रत्येक एड्रैस फील्ड (Address Field) या तो प्रोसेसर रजिस्टर (Processor Register) या मेमोरी ऑपरेन्ड (Memory Operand) को प्रदर्शित कर सकती है। असेम्बली लैंग्वेज (Assembly Language) में लिखा गया प्रोग्राम जो $X = (A + B) * (C + D)$ को हल करता है, उसे नीचे प्रदर्शित किया गया है। यहाँ प्रत्येक निर्देश (Instruction) के साथ रजिस्टर ट्रांसफर (Register Transfer) को कमेन्ट्स (Comments) के द्वारा प्रदर्शित किया गया है—

```
ADD  R1, A, B    R1 ← M[A] + M[B]
ADD  R2, C, D    R2 ← M[C] + M[D]
MUL  X, R1, R2   M[X] ← R1 * R2
```

यहाँ यह माना गया है कि कम्प्यूटर में दो प्रोसेसर रजिस्टर $R1$ और $R2$ का प्रयोग किया गया है। चिह्न $M[A]$, A द्वारा चिह्नित एड्रैस (Address) पर सुरक्षित मेमोरी वर्ड (Memory Word) को प्रदर्शित कर रहा है।

थ्री-एड्रैस इन्सट्रक्शन फॉरमेट (Three-Address Instruction Format) का लाभ यह है कि जब अर्थमेटिक-एक्सप्रेशन (Arithmetic Expression) को हल करने का प्रोग्राम बनाया जाता है, तो वो काफी छोटा होता

Scanned with CamScanner

है। इसकी हानि यह है कि तीन एड्रैसेज (Addresses) को प्रदर्शित करने के लिये बायनरी-कोडेड इन्सट्रक्शन (Bi
Coded Instruction) को अनेक बिट्स (Bits) की आवश्यकता पड़ती है। सायबर 170 (Cyber 170) वह क
है, जो थ्री-एड्रैस इन्सट्रक्शन्स (Three-Address Instructions) का प्रयोग करता है।

दो-एड्रैस वाले निर्देश Two-Address Instructions सामान्य व्यावसायिक कम्प्यूटर्स में टू-एड्रैस इन्सट्र
(Two-Address Instructions) का मुख्यत: प्रयोग किया जाता है। यहाँ भी प्रत्येक एड्रैस फील्ड (Address fiel
तो प्रोसेसर रजिस्टर (Processor Register) या मेमोरी वर्ड (Memory Word) को प्रदर्शित कर सक
$X = (A + B) * (C + D)$ को हल करने के लिए प्रोग्राम निम्नानुसार है—

```
MOV    R1, A     R1 ← M[A]
ADD    R1, B     R1 ← R1 + M[B]
MOV    R2, C     R2 ← M[C]
ADD    R2, D     R2 ← R2 + M[D]
MUL    R1, R2    R1 ← R1 * R2
MOV    X, R1     M[X] ← R1
```

MOV निर्देश (Instruction) के द्वारा ऑपरेन्ड्स (Operands) का मेमोरी (Memory) से आदान-प्रदान किया
है। एक निर्देश में दिये गये पहले चिह्न को स्रोत (Source) और गंतव्य (Destination) दोनों समझा जाता है
ऑपरेशन (Operation) के उत्तर को ट्रान्सफर (Transfer) किया जाता है।

एक-एड्रैस वाले निर्देश One-Address Instructions वन-एड्रैस इन्सट्रक्शन (One Address Instruction)
मैनीपुलेशन (Data Manipulation) के लिए एक्यूमुलेटर रजिस्टर (Accumulator Register-*AC*) का प्रयोग
है। गुणा और भाग के लिए दूसरे रजिस्टर की आवश्यकता होती है। फिर भी हम यह मान कर चलते हैं कि दूसरे र
आवश्यक नहीं हैं और *AC* में सभी ऑपरेशन्स (Operations) के उत्तर या परिणाम को सुरक्षित किया
$X = (A + B) * (C + D)$ को हल करने के लिए प्रोग्राम निम्नलिखित है—

```
LOAD   A    AC ← M[A]
ADD    B    AC ← AC + M[B]
STORE  T    M[T] ← AC
LOAD   C    AC ← M[C]
ADD    D    AC ← AC + M[D]
MUL    T    AC ← AC * M[T]
STORE  X    M[X] ← AC
```

यहाँ सभी ऑपरेशन्स (Operations) मेमोरी ऑपरेन्ड (Memory Operand) और *AC* रजिस्टर के मध्य कि
हैं। *T* टेम्पररी मेमोरी (Temporary Memory) का एड्रैस (Address) है, जहाँ इण्टरमीडिएट (Intermed
परिणाम को सुरक्षित किया जाता है।

शून्य-एड्रैस वाले निर्देश Zero-Address Instructions स्टैक-ऑर्गेनाइज्ड (Stack Organised) कम्प्यूटर में
(Instruction) ADD और MUL के लिये किसी एड्रैस फील्ड (Address Field) की आवश्यकता नहीं हो
हालांकि Push और Pop निर्देशों (Instructions) को एक एड्रैस (Address) की आवश्यकता होती है, जो
(Stack) के साथ संचार करने वाले ऑपरेन्ड (Operand) को प्रदर्शित करता है। स्टैक ऑर्गेनाइज्ड (S
Organised) कम्प्यूटर के द्वारा समीकरण $X = (A + B) * (C + D)$ को हल करने का प्रोग्राम निम्नलिखित है
प्रोग्राम में स्टैक (Stack) के टॉप (Top) को TOS (Top of Stack) से प्रदर्शित किया गया है—

```
PUSH   A   TOS ← A
PUSH   B   TOS ← B
ADD        TOS ← (A + B)
PUSH   C   TOS ← C
PUSH   D   TOS ← D
```

Scanned with CamScanner

```
ADD     TOS ← (C + D)
MUL     TOS ← (C + D) * (A + B)
POP   X  M[X] ← TOS
```

स्टैक कम्प्यूटर (Stack Computer) में एक अर्थमेटिक एक्सप्रेशन (Arithmetic Expression) को हल करने के लिए, समीकरण को रिवर्स पोलिश नोटेशन (Reverse Polish Notation) के रूप में निरूपित करने की आवश्यकता होती है।

प्रश्न 9. एड्रैसिंग मोड (Addressing Mode) से आप क्या समझते हैं? इसके चरण (Steps) एवं उपयोगिता को समझाइए।

उत्तर **एड्रैसिंग मोड** Addressing mode किसी भी निर्देश (Instruction) की ऑपरेशन फील्ड (Operation Field) यह बताती है कि कौन-सा ऑपरेशन (Operation) किया जाना है। दिये गये ऑपरेशन (Operation) को कम्प्यूटर रजिस्टर (Register) अथवा मेमोरी वर्ड (Memory Word) में सुरक्षित डेटा पर एग्जिक्यूट (Execute) किया जाता है।

प्रोग्राम एग्जिक्यूशन (Program Execution) के दौरान किस प्रकार ऑपरेन्ड (Operand) को चुनना है, यह निर्देश (Instruction) के एड्रैसिंग मोड (Addressing Mode) पर निर्भर करता है। एड्रैसिंग मोड्स (Addressing Modes), एक नियम को परिभाषित करते हैं, जिनके द्वारा ऑपरेन्ड (Operand) का रेफरेन्स (Reference) करने से पहले निर्देश (Instruction) की एड्रैस फील्ड (Address Field) को इन्टरप्रेट (Interpret) अथवा मॉडीफाई (Modify) किया जाता है।

कम्प्यूटर्स (Computers) में निम्नलिखित एक अथवा दोनों उद्देश्यों को पूर्ण करने के लिए एड्रैसिंग मोड्स (Addressing Modes) का उपयेग किया जाता है।

1. मेमोरी (Memory) को प्वॉइन्टर (Pointer), लूप कण्ट्रोल (Loop Control) के लिए काउन्टर्स (Counters), डेटा की इन्डैक्सिग (Data Indexing) और प्रोग्राम रिलोकेशन (Program Relocation) के द्वारा प्रोयागकर्त्ता (User) को प्रोग्रामिंग वैसेटिलिटि (Programming Versatility) प्रदान करना।
2. निर्देश (Instruction) की एड्रैसिंग फील्ड (Addressing Field) में बिट्स (Bits) की संख्या को कम करना।

विभिन्न एड्रैसिंग मोड्स (Addressing Modes) को समझने के लिए, पहले हमें कम्प्यूटर की मूल ऑपरेशन साइकल (Basic Operation Cycle) को समझना होगा। कम्प्यूटर की कण्ट्रोल यूनिट (Control Unit) को इन्सट्रक्शन साइकल (Instruction Cycle) द्वारा कार्य करने के लिए डिजाइन (Design) किया गया है। इन्सट्रक्शन साइकल (Instruction Cycle) मुख्यत: निम्नलिखित तीन चरणों में विभाजित होती है—

1. निर्देश (Instruction) को मेमोरी (Memory) से फैच (Fetch) करें।
2. निर्देश (Instruction) को डिकोड (Decode) करें।
3. निर्देश (Instruction) को एग्जिक्यूट (execute) करें।

चरण 1 (Phase I) कम्प्यूटर में एक ऐसा रजिस्टर होता है जो मेमोरी (Memory) में सुरक्षित प्रोग्राम (Program) के निर्देशों (Instruction) का ट्रैक (Track) बनाये रखता है। इस रजिस्टर को प्रोग्राम काउन्टर (Program Counter) अथवा *PC* कहा जाता है। *PC* में एग्जिक्यूट (Execute) किये जाने वाले अगले निर्देश (Instruction) का एड्रैस (Address) होता है। प्रत्येक समय जब भी एक निर्देश (Instruction) को मेमोरी (Memory) से फैच (Fetch) किया जाता है, तब *PC* के मान में इन्क्रीमेन्ट (Increment) कर दिया जाता है।

चरण 2 (Phase II) इस चरण में किये जाने वाले ऑपरेशन (Operation), निर्देश (Instruction) के एड्रैसिंग मोड (Addressing Mode) और ऑपरेन्ड (Operand) की लोकेशन (Location) का निर्धारण किया जाता है।

चरण 3 (Phase III) अन्तत: निर्देश (Instruction) को एग्जिक्यूट (Execute) कर दिया जाता है।

कम्प्यूटर पुन: चरण 1 पर पहुँचकर श्रेणी के अगले निर्देश पर कार्य करता है।

Scanned with CamScanner

कुछ कम्प्यूटर में एड्रैसिंग मोड (Addressing Mode) को ऑपरेशन कोड (Operation Code) की भाँति ही पृथक बायनरी कोड (Distinct Binary Code) द्वारा बताया जाता है। कुछ कम्प्यूटर्स में एक ही बायनरी कोड (Binary Code) का प्रयोग ऑपरेशन कोड (Operation Code) और एड्रैसिंग मोड (Addressing Mode), दोनों को बतलाने के लिए किया जाता है। निर्देश (Instruction) को विभिन्न प्रकार के एड्रैसिंग मोड्स (Addressing Modes) द्वारा परिभाषित किया जा सकता है और कभी-कभी एक से अधिक एड्रैसिंग मोड्स (Addressing Modes) को एक ही निर्देश (Instruction) में कम्बाइन (Combine) किया जाता है।

पृथक् एड्रैसिंग मोड फील्ड (Distinct Addressing Mode Field) वाले इन्स्ट्रक्शन फॉरमेट (Instruction Format) को निम्नांकित चित्र 1.4 में दर्शाया गया है—

Opcode	Mode	Address

चित्र 1.4 Instruction Format with a Distinct Addressing Mode Field

यहाँ Opcode अर्थात् ऑपरेशन कोड (Operation Code) बताता है कि कौन-सा ऑपरेशन किया जाना है। Mode फील्ड, ऑपरेशन (Operation) के लिए आवश्यक ऑपरेन्ड्स (Operands) को लोकेट (Locate) करने के लिए उपयोग की जाती है। Address फील्ड का होना आवश्यक नहीं है, यह हो भी सकती है और नहीं भी। यदि एड्रैस फील्ड (Address Field) है, तो इसका तात्पर्य है कि यह मेमोरी एड्रैस (Memory Address) अथवा प्रोसेसर रजिस्टर (Processor Register) को दर्शा सकती हैं जैसा कि हम जानते हैं कि एक निर्देश में एक अथवा अधिक एड्रैस फील्ड्स हो सकती है और प्रत्येक फील्ड (Field) अपने विशेष एड्रैसिंग मोड (Addressing Mode) से जुड़ी हो सकती है।

उपलब्ध विभिन्न एड्रैसिंग मोड्स (Addressing Modes) में से दो मोड्स (Modes) ऐसे हैं, जिनमें किसी एड्रैसिंग फील्ड की आवश्यकता नहीं होती है। ये (Modes) हैं—इम्प्लाइड (Implied) और इमीडिएट (Immediate)।

प्रश्न 10. इम्प्लाइड मोड (Implied Mode), इमिडिएट मोड (Immediate Mode) तथा रजिस्टर मोड (Register Mode) को समझाइए।

उत्तर **इम्प्लाइड मोड** Implied Mode इस मोड (Mode) में ऑपरेन्ड्स (Operands). निर्देश (Instruction) में आन्तरिक रूप से परिभाषित होते हैं। उदाहरण के लिए, कॉम्प्लीमेन्ट एक्यूमुलेटर (Complement Accumulator), एक इम्प्लाइड मोड निर्देश (Implied Mode Instruction) है, क्योंकि निर्देश (Instruction) की परिभाषा में एक्यूमुलेटर (Accumulator) में ऑपरेन्ड (Operand) आन्तरिक रूप से दिया गया है। इस प्रकार हम कह सकते हैं कि वे सभी रजिस्टर रेफरेन्स निर्देश (Register Reference Instructions), जो एक्यूमुलेटर (Accumulator) का प्रयोग करते हैं, उन्हें इम्प्लाइड मोड निर्देश (Implied Mode Instruction) कहा जा सकता है। स्टैक ऑर्गेनाइज्ड (Stack Organised) कम्प्यूटर के सभी जीरो-एड्रैस निर्देश (Zero Address Instruction), इम्प्लाइड मोड निर्देश (Implied Mode Instruction) हैं, क्योंकि ऑपरेन्ड्स (Operands) स्टैक (Stack) के टॉप (Top) पर अन्तर्निहित होते हैं।

इमिडिएट मोड Immediate mode इस प्रकार के मोड (Mode) में ऑपरेन्ड (Operand) निर्देश में ही दिये होते हैं। दूसरे शब्दों में इस प्रकार के मोड (Mode) के निर्देशों (Instructions) में एड्रैस फील्ड (Address Field) के स्थान पर ऑपरेन्ड फील्ड (Operand Field) ही दी गयी होती है। ऑपरेन्ड फील्ड (Operand Field), वास्तविक ऑपरेन्ड (Operand) है, जिस पर निर्देश (Instruction) में दिये गये ऑपरेशन (Operation) को एग्जिक्यूट (Execute) कराना है। इमीडिएट मोड निर्देशों (Immediate Mode Instructions) का प्रयोग रजिस्टर्स (Registers) को कॉन्सटेन्ट वैल्यू (Constant Value) में इनिशिएलाइज (Initialize) करने के लिए किया जाता है।

रजिस्टर मोड Register Mode हम पढ़ चुके हैं कि एक निर्देश (Instruction) की एड्रैस फील्ड (Address Field), मेमोरी वर्ड (Memory Word) अथवा प्रोसेसर रजिस्टर (Processor Register), किसी को भी प्रदर्शित कर सकती है। जब एड्रैस फील्ड एक प्रोसेसर रजिस्टर को प्रदर्शित करती है, तो निर्देश (Instruction), रजिस्टर मोड (Register Mode) में होता है।

Scanned with CamScanner

इस मोड (Mode) में ऑपरेन्ड्स (Operands) सीपीयू (CPU) के रजिस्टर्स (Registers) में होते हैं। निर्देश (Instruction) में दी गयी रजिस्टर फील्ड (Register Field) के द्वारा विशेष रजिस्टर (Register) को सेलेक्ट (Select) किया जाता है। जैसा कि हम पढ़ चुके हैं कि R बिट (Bit) के एड्रैस (Address) द्वारा 2^R रजिस्टर्स (Registers) में से किसी भी एक को सेलेक्ट (Select) किया जा सकता है।

प्रश्न 11. रजिस्टर इनडायरेक्ट मोड (Register Indirect Mode) एवं आटोइनक्रीमेन्ट मोड (Autoincrement Mode) को परिभाषित कीजिए।

उत्तर **रजिस्टर इनडायरेक्ट मोड** Register Indirect Mode इस मोड (Mode) में निर्देश (Instruction), सीपीयू (CPU) के एक रजिस्टर को उल्लेखित अर्थात् स्पेसीफाई (Specify) करता है और इस रजिस्टर में मेमोरी (Memory) में दिये गये ऑपरेन्ड (Operand) का एड्रैस (Address) दिया होता है। दूसरे शब्दों में, चुने गये रजिस्टर में ऑपरेन्ड (Operand) न होकर ऑपरेन्ड (Operand) का एड्रैस (Address) दिया होता है। रजिस्टर इनडायरेक्ट मोड निर्देश (Register Indirect Mode Instruction) को प्रयोग करने से पहले प्रोग्रामर (Programmer) को यह सुनिश्चित कर लेना चाहिए कि पिछले निर्देश (Instruction) द्वारा ऑपरेन्ड (Operand) का एड्रैस (Address), प्रोसेसर रजिस्टर (Processor Register) में भेज दिया गया है।

रजिस्टर इनडायरेक्ट मोड निर्देश (Register Indirect Mode Instruction) का मुख्य लाभ यह है कि एक रजिस्टर का रेफरेन्स (Reference) देने के लिए निर्देश (Instruction) की एड्रैस फील्ड्स (Address Fields) की कुछ ही बिट्स का प्रयोग किया जाता है, जबकि एक मेमोरी वर्ड (Memory Word) का डायरेक्ट रेफरेन्स (Direct Reference) देने के लिए तुलनात्मक रूप में अधिक बिट्स (Bits) की आवश्यकता होगी।

आटोइनक्रीमेन्ट मोड Autoincrement Mode यह रजिस्टर इनडायरेक्ट मोड (Register Indirect Mode) के समान ही है। बस अन्तर यह है कि इसमें मेमोरी (Memory) को एक्सेस (Access) करने के लिए रजिस्टर का प्रयोग करने के पश्चात् अथवा प्रयोग करने से पहले, इसके मान में इनक्रीमेन्ट (Increment) अथवा डिक्रीमेन्ट (Decrement) किया जाता है। जब रजिस्टर (Register) में सुरक्षित एड्रैस (Address) किसी डेटा टेबल (Data Table) को रेफर (refer) करता है, तो डेटा टेबल (Data Table) के प्रत्येक एक्सेस (Access) के पश्चात् रजिस्टर में इनक्रीमेन्ट (Increment) अथवा डिक्रीमेन्ट (Decrement) करना आवश्यक हो जाता है। इसको इनक्रीमेन्ट (Increment) अथवा डिक्रीमेन्ट (Decrement) निर्देशों (Instructions) की सहायता से किया जा सकता है।

सीपीयू (CPU) में कण्ट्रोल यूनिट (Control Unit), एक निर्देश (Instruction) की एड्रैस फील्ड (Address Field) का प्रयोग मेमोरी (Memory) से ऑपरेन्ड (Operand) को प्राप्त करने के लिए करती है। कभी-कभी एड्रैस फील्ड (Address Field) में दिया गया मान ऑपरेन्ड (Operand) के एड्रैस को प्रदर्शित करता है और कभी-कभी यह वो एड्रैस (Address) होता है, जिसके द्वारा ऑपरेन्ड (Operand) के एड्रैस (Address) की गणना की जाती है। विभिन्न एड्रैसिंग मोड्स (Addressing Modes) को पहचानने के लिए यह आवश्यक है कि निर्देश (Instruction) के एड्रैस भाग (Address Part) और निर्देश (Instruction) को एग्जिक्यूट करते समय कण्ट्रोल (Control) के द्वारा प्रयोग किये जाने वाले प्रभावी एड्रैस (Effective Address) में अन्तर किया जाए। किये गये एड्रैसिंग मोड (Addressing Mode) के द्वारा बतायी गयी गणना करने के बाद प्राप्त मेमोरी एड्रैस (Memory Address) को प्रभावी अर्थात् इफेक्टिव एड्रैस (Effective Address) कहा जाता है। कम्प्यूटेशनल टाइप निर्देश (Computational Type Instruction) में ऑपरेन्ड (Operand) का एड्रैस (Address) इफेक्टिव एड्रैस (Effective Address) होता है।

प्रश्न 12. इफेक्टिव एड्रैस (Effective Address) क्या होता है? इसके प्रकारों का वर्णन कीजिए।

उत्तर **इफेक्टिव एड्रैस** Effective Address इफेक्टिव एड्रैस, वह एड्रैस होता है, जहाँ एक ब्रान्च-टाइप निर्देश (Branch Type Instruction) की प्रतिक्रिया में कण्ट्रोल ब्रान्चेज (Control Branches) होती हैं। इस प्रकार के दो एड्रैसिंग मोड्स (Addressing Modes) होते हैं—

Scanned with CamScanner

(i) **डायरेक्ट एक्सेस मोड** Direct Access Mode इस मोड (Mode) में इफेक्टिव एड्रैस (Effective Address) का मान निर्देश (Instruction) के एड्रैस भाग (Address Part) के मान के समान होता है। ऑपरेन्ड (Operand) मेमोरी में होता है और इसका एड्रैस (Address), निर्देश (Instruction) की एड्रैस फील्ड (Address Field) में दिया जाता है। ब्रान्च टाइप इन्सट्रक्शन (Branch Type Instruction) में एड्रैस फील्ड (Address Field), वास्तविक ब्राँच एड्रैस (Branch Address) को परिभाषित करती है।

(ii) **इनडायरेक्ट एक्सेस मोड** Indirect Access Mode इस मोड (Mode) में निर्देश (Instruction) की एड्रैस फील्ड (Address Field), उस एड्रैस (Address) को धारण करती है, जहाँ मेमोरी (Memory) में इफेक्टिव एड्रैस (Effective Address) को सुरक्षित अथवा स्टोर (Store) किया गया है। कण्ट्रोल (Control), मेमोरी (Memory) से एड्रैस (Address) को फैच (Fetch) करती है और इसके एड्रैस (Address) भाग के द्वारा पुनः मेमोरी एक्सेस (memory Access) करके इफेक्टिव एड्रैस (Effective Address) को प्राप्त कर लेती है।

कुछ एड्रैसिंग मोड्स (Addressing Modes) में निर्देश (Instruction) की एड्रैस फील्ड (Address Field) को सीपीयू (CPU) के विशेष रजिस्टर के मान के साथ जोड़ा जाता है। इस प्रकार इन मोड्स (Modes) में इफेक्टिव एड्रैस (Effective Address) की गणना निम्नलिखित प्रकार से की जा सकती है—

Effective Address = Address Part of Instruction + Content of CPU Register

गणना में प्रयुक्त किया गया CPU रजिस्टर, प्रोग्राम काउन्टर (Program Counter), इन्डेक्स रजिस्टर (Index Register) अथवा बेस रजिस्टर (Base Register) हो सकता है। किसी भी परिस्थिति में हमें एक भिन्न एड्रैसिंग मोड (Addressing Mode) प्राप्त होता है, वह एक भिन्न एप्लीकेशन के लिए प्रयोग किया जाता है।

प्रश्न 13. रिलेटिव एड्रैस मोड (Relative Address Mode) तथा इन्डैक्स्ड एड्रैसिंग मोड (Indexed Addressing Mode) का वर्णन कीजिए।

उत्तर **रिलेटिव एड्रैस मोड** Relative Address Mode इस मोड (Mode) में इफेक्टिव एड्रैस (Effective Address) को गणना करने के लिए निर्देश (Instruction) के एड्रैस पार्ट (Address Part) में प्रोग्राम काउन्टर (Program Counter) के कन्टेन्ट (Content) को जोड़ा जाता है। सामान्यत: निर्देश का एड्रैस पार्ट (Address Part) एक साइनड संख्या (Signed Number), 2's काम्पलीमेन्ट (2's Complement) में निरूपित है, जो धनात्मक (Positive) अथवा ऋणात्मक (Negative) हो सकती हैं। जब इस संख्या को प्रोग्राम काउन्टर (Program Counter) के कन्टेन्ट (Content) के साथ जोड़ा जाता है, तो हमें इफेक्टिव एड्रैस (Effective Address) प्राप्त होता है। इस इफेक्टिव एड्रैस (Effective Address) का मेमोरी (Memory) में स्थान, अगले निर्देश के एड्रैस से सम्बन्धित अर्थात् रिलेटिव (Relative) होगा। इसे समझने के लिए हम एक उदाहरण की सहायता लेंगे। मान लेते हैं कि प्रोग्राम काउन्टर (Program Counter) में संख्या 740 है और निर्देश के एड्रैस पार्ट (Address Part) में संख्या 30 है। अब लोकेशन (Location) 740 पर दिये गये निर्देश (Instruction) को मेमोरी (Memory) से फैच (Fetch) किया जाता है और तत्पश्चात् इनक्रीमेन्ट (Increment) होकर प्रोग्राम काउन्टर (Program Counter) का मान 741 हो जाता है। रिलेटिव एड्रैस मोड (Relative Address Mode) की गणना के अनुसार प्राप्त इफेक्टिव एड्रैस (Effective Address) 741 + 30 = 771 है। यह अगले निर्देश (Instruction) के एड्रैस (Address) से 30 मेमोरी लोकेशन (Memory Location) आगे है। इस प्रकार के एड्रैसिंग मोड (Addressing Mode) में कम बिट्स (Bits) का उपयोग किया जाता है।

इन्डैक्स्ड एड्रैसिंग मोड Indexed Addressing Mode इस मोड (Mode) में इफेक्टिव एड्रैस (Effective Address) को प्राप्त करने के लिए निर्देश (Instruction) के एड्रैस पार्ट (Address Part) में इन्डैक्स रजिस्टर (Index Register) के कन्टेन्ट (Content) को जोड़ा जाता है। इन्डैक्स रजिस्टर (Index Register) सीपीयू (CPU) का एक विशेष रजिस्टर है, जिसमें इन्डैक्स (Index) मान को सुरक्षित किया जाता है। निर्देश (Instruction) का एड्रैस पार्ट (Address Part) मेमोरी (Memory) में डेटा ऐरे (Data Array) के आरम्भ के एड्रैस (Address) से सम्बन्धित अर्थात् रिलेटिव (Relative) होकर सुरक्षित होता है। आरम्भ के एड्रैस (Beginning Address) और ऑपरेन्ड (Operand) के एड्रैस के अन्तर को इन्डैक्स मान (Index Value) कहा जाता है और इसे इन्डैक्स रजिस्टर (Index Register) में सुरक्षित किया जाता है। ऐरे (Array) के किसी भी ऑपरेन्ड (Operand) को एक ही निर्देश (Instruction) के द्वारा प्राप्त किया

Scanned with CamScanner

जा सकता है। हमें बस सही इन्डैक्स मान (Index Value) की आवश्यकता होती है। यदि हमें एक ऐरे के सभी अथवा कुछ कन्सिक्यूटिव ऑपरेन्ड्स (Consecutive Operands) का प्रयोग करना हो, तो इन्डैक्स रजिस्टर (Index Register) को इनक्रीमेन्ट (Increment) किया जा सकता है। ध्यान दें कि यदि इन्डैक्स-टाइप निर्देश (Index Type Instruction) में एड्रैस फील्ड नहीं दी गयी हो, तो निर्देश (Instruction) ऑपरेशन के रजिस्टर इनडायरेक्ट मोड (Register Indirect Mode) में परिवर्तित हो जाता है।

प्रश्न 14. बेस रजिस्टर एड्रैसिंग मोड (Base Register Addressing Mode) को समझाइए।

उत्तर **बेस रजिस्टर एड्रैसिंग मोड** Base Register Addressing Mode इस मोड (Mode) में इफैक्टिव एड्रैस (Effective Address) को प्राप्त करने के लिए निर्देश (Instruction) के एड्रैस पार्ट (Address Part) में बेस रजिस्टर (Base Register) के कन्टेन्ट (Content) को जोड़ा जाता है। यह इन्डैक्स्ड एड्रैसिंग मोड (Indexed Addressing Mode) के समान ही होता है, बस अन्तर यह है कि प्रयोग किया जाने वाला रजिस्टर इन्डैक्स (Index) न होकर बेस रजिस्टर (Base Register) होता है। इन दोनों के कार्य अर्थात् गणना करने का तरीका भिन्न होता है। यह इन्डैक्स रजिस्टर (Index Register) में इन्डैक्स संख्या (Index Number) सुरक्षित होती है, जिसे ऐरे (Array) के बिगनिंग एड्रैस (Beginning Address) में जोड़ा जाता है अर्थात् यह मान निर्देश (Instruction) के एड्रैस (Address) भाग से सम्बन्धित अर्थात् रिलेटिव (Relative) होता है। बेस रजिस्टर (Base Register) में बेस एड्रैस (Base Address) सुरक्षित होता है और निर्देश (Instruction) के एड्रैस पार्ट (Address Part) में बेस एड्रैस (Base Address) से जुड़ी डिसप्लेसमेन्ट (Displacement) सुरक्षित होती है। यहाँ हमें ऐरे (Array) के विभिन्न ऑपरेन्ड्स (Operands) को फैच (Fetch) करने के लिए विभिन्न निर्देशों (Instruction) का उपयोग करना पड़ता है।

प्रश्न 15. कण्ट्रोल यूनिट को कितने प्रकार से व्यवस्थित किया जा सकता है?

उत्तर **कण्ट्रोल यूनिट ऑर्गेनाइजेशन** Control Unit Organisation कण्ट्रोल यूनिट (Control Unit) को दो प्रकार से व्यवस्थित (Organise) किया जा सकता है—हार्डवायर्ड कण्ट्रोल (Hardwired Control) और माइक्रोप्रोग्राम्ड कण्ट्रोल (Microprogrammed Control)।

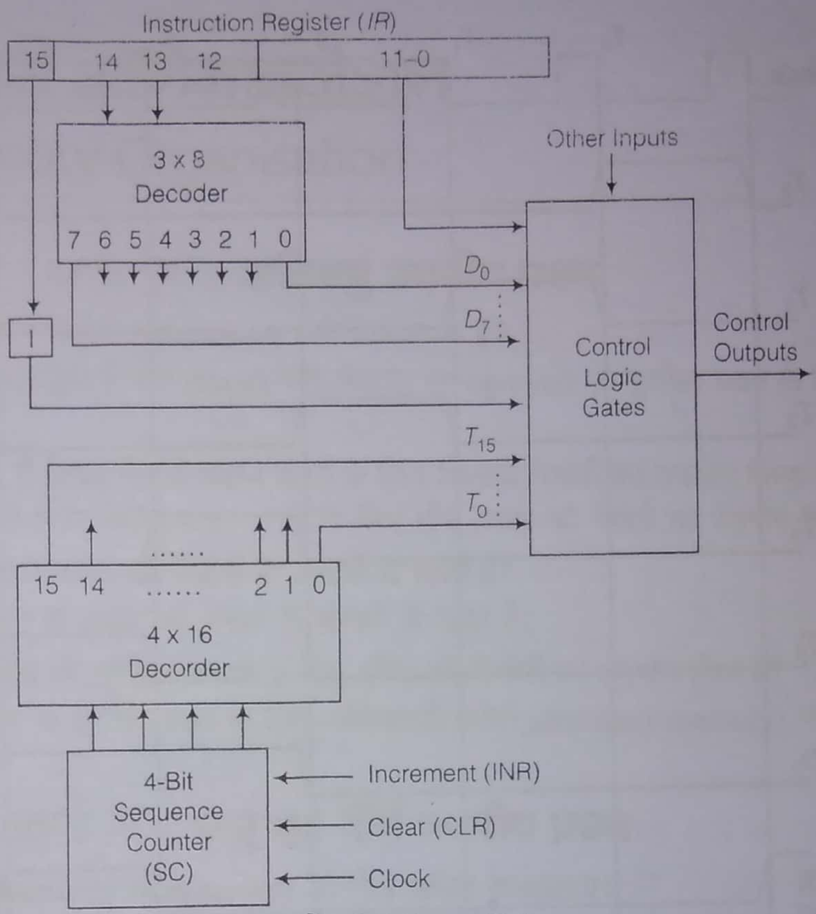
1. **हार्डवायर्ड कण्ट्रोल** Hardwired Control हार्डवायर्ड ऑर्गेनाइजेशन (Hardwired Organisation) में कण्ट्रोल लॉजिक (Control Logic) को गेट्स (Gates), फ्लिप-फ्लाप्स (Flip-Flops), डिकोडर (Decoder) और अन्य डिजिटल सर्किट्स (Digital Circuits) द्वारा कार्यान्वित किया जाता है। इसका लाभ यह है कि इसके द्वारा ऑपरेशन की अत्यधिक गति को प्राप्त किया जा सकता है। इसमें यदि डिजाइन में परिवर्तन होता है, तो विभिन्न तत्त्वों के मध्य की वायरिंग (Wiring) को परिवर्तित करना पड़ता है।
2. **माइक्रोप्रोग्राम्ड कण्ट्रोल** Microprogrammed Control माइक्रोप्रोग्राम्ड ऑर्गेनाइजेशन (Microprogrammed Organisation) में कण्ट्रोल की सूचना को कण्ट्रोल मेमोरी (Control Memory) में सुरक्षित किया जाता है। कण्ट्रोल मेमोरी को माइक्रोऑपरेशन की आवश्यक श्रेणी को प्रारम्भ करने के लिए प्रोग्राम किया जाता है। मल्टीप्रोग्राम्ड कण्ट्रोल में कोई भी वांछित परिवर्तन करने के लिए कण्ट्रोल मेमोरी के माइक्रोप्राग्राम्ड को अपडेट (Update) किया जाता है।

प्रश्न 16. हार्डवायर्ड कण्ट्रोल यूनिट विस्तारपूर्वक समझाइए।

उत्तर **हार्डवायर्ड कण्ट्रोल यूनिट** Hardwired Control Unit इस प्रकार की कण्ट्रोल यूनिट (Control Unit) को चित्र 1.5 में प्रदर्शित किया गया है—

इसमें दो डिकोडर्स (Decoders), एक सिक्वेन्स काउन्टर (Sequence Counter) और विभिन्न कण्ट्रोल लॉजिक गेट्स (Control Logic Gates) का उपयोग किया गया है। मेमोरी (Memory) से रीड़ (Read) किये गये निर्देश (Instruction) को इन्सट्रक्शन रजिस्टर (Instruction Register) में सुरक्षित किया जाता है। यहाँ इन्सट्रक्शन रजिस्टर (Instruction Register) को तीन भागों में विभाजित किया गया है—1 बिट ऑपरेशन कोड (Operation Code) और 0 से 11 बिट्स। बिट्स 12 से 14 में दिये गये ऑपरेशन कोड को 3×8 डिकोडर (3×8 Decoder) के द्वारा डिकोड (Decode) किया जाता है। डिकोडर के आठों आउटपुट्स को D_0 से D_7 तक के चिह्नों द्वारा निरूपित किया गया है। सब्स्क्रिप्ट में दिए गए डेसीमल संख्या का मान उस आपरेशन कोड के बायनरी मान के समान होता है। निर्देश

Scanned with CamScanner



चित्र **1.5** Control Unit

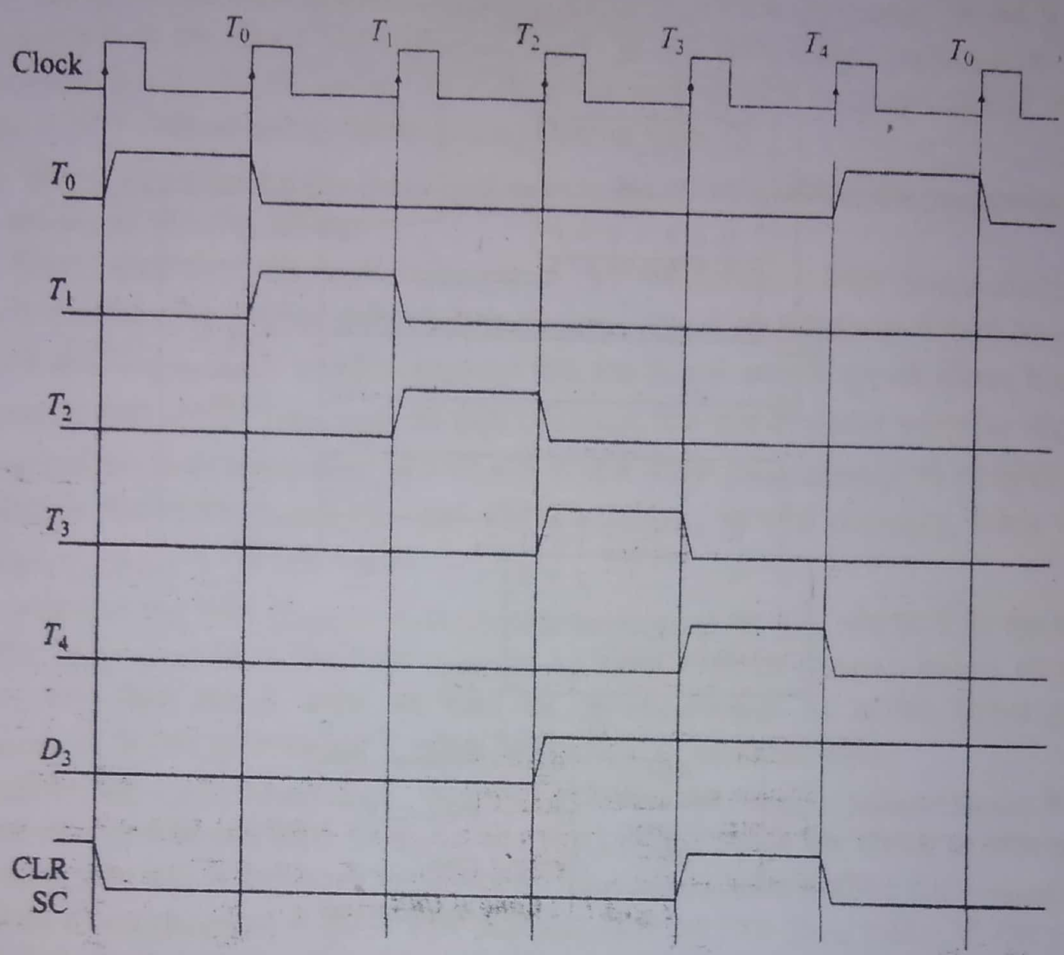
(Instruction) की 15 बिट को 1 द्वारा चिह्नित फ्लिप-फ्लाप (Flip-Flop) में प्रवाहित किया जाता है। 0 से 11 की बिट्स को कण्ट्रोल लॉजिक गेट्स पर प्रवाहित किया जाता है। 4-बिट के सिक्वेन्स काउन्टर (Sequence Counter) का उपयोग 0 से 15 तक की बायनरी गणना के लिए किया जा सकता है। सिक्वेन्स काउन्टर के आउटपुट्स को डिकोडर पर प्रवाहित कर 16 टाइमिंग सिगनल (Timing Signal) T_0 से T_{15} तक में डिकोड (Decode) किया जाता है। *SC* अर्थात् सिक्वेन्स काउन्टर (Sequence Counter) को समय के अनुसार बढ़ाया और घटाया जा सकता है। अधिकतर, सिक्वेन्स काउन्टर में इनक्रीमेन्ट का उपयोग टाइमिंग सिगनल्स (Timing Signals) की एक श्रेणी (Sequence) को उत्पन्न करने के लिए किया जाता है। जब काउन्टर का मान 0 कर दिया जाता है, तो टाइमिंग सिगनल प्रारम्भ अर्थात् T_0 से शुरू हो जाता है। उदाहरण के लिए, मान लेते हैं कि *SC* को T_0, T_1, T_2, T_3 और T_4 टाइमिंग सिगनल उत्पन्न करने के लिए इन्क्रीमेन्ट किया जायेगा। समय T_4 पर यदि डिकोडर आउटपुट D_3 सक्रिय है, तो *SC* को क्लिअर (Clear) करके 0 कर दिया जायेगा। इसको चिह्नों द्वारा निम्नलिखित रूप में निरूपित किया जा सकता है—

$$D_3T_4 : \quad SC \leftarrow 0$$

चित्र 1.6 में दर्शाये गये टाइमिंग डायग्राम (Timing Diagram) में हमने कण्ट्रोल सिगनल के समय सम्बन्ध (Time Relationship) को प्रदर्शित किया है।

सिक्वेन्स काउन्टर (Sequence Counter) *SC* क्लॉक के पॉजिटिव ट्रान्जिशन (Positive Transition) पर प्रतिक्रिया करता है। प्रारम्भ में, *SC* का *CLR* इनपुट सक्रिय होता है। क्लॉक का पहला पॉजिटिव ट्रान्जिशन (Positive Transition) *SC* को 0 करता है, जिसके फलस्वरूप डिकोडर (Decoder) से टाइमिंग सिगनल (Timing Signal) T_0 सक्रिय हो जाता है। एक क्लॉक सायंकाल के दौरान T_0 सक्रिय है। अब T_0 द्वारा चिह्नित पॉजिटिव क्लॉक ट्रान्जिशन (Positive Clock Transition) पर केवल उन्हीं रजिस्टर को ट्रिगर (Trigger) किया जायेगा, जिनके कण्ट्रोल

Scanned with CamScanner

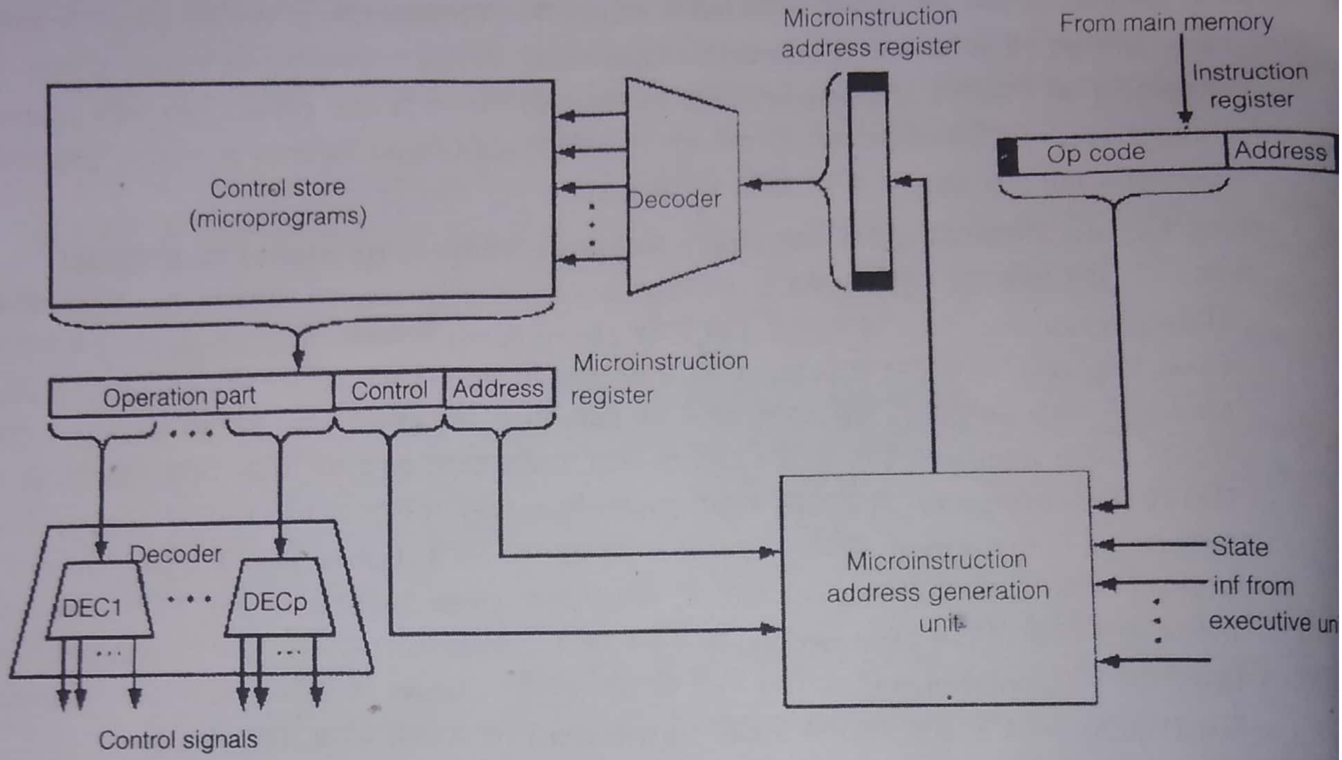


चित्र **1.6** Example of Control Timing Signals

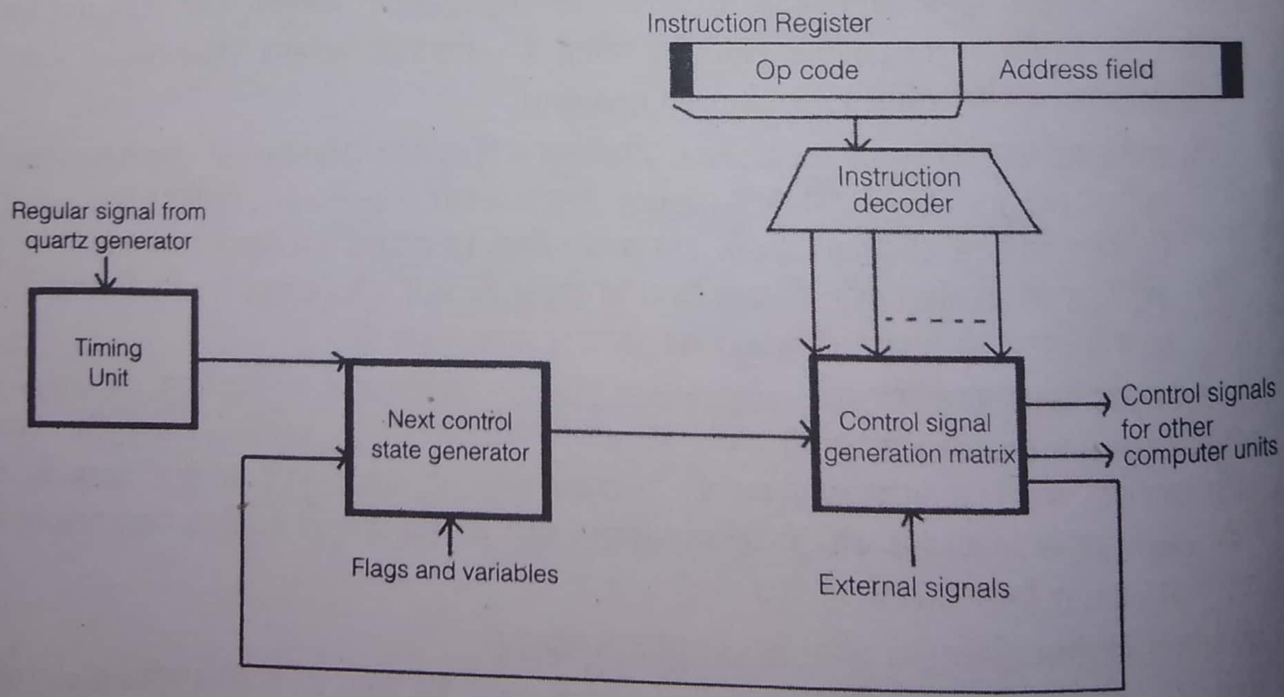
(Control) इनपुट टाइमिंग सिगनल (Input Timing Signal) T_0 से जुड़े हुए होंगे। जब तक SC में CLR सक्रिय है, तब तक प्रत्येक पॉजिटिव क्लॉक ट्रान्जिशन पर SC को इन्क्रीमेन्ट किया जाता है। इसके द्वारा एक श्रेणी (Sequence) में टाइमिंग सिगनल (Timing Signal) को उत्पन्न किया जाता है—T_0, T_1, T_2, T_3, T_4 और आगे, यदि SC को बीच में 0 नहीं किया जाता तो टाइमिंग सिगनल ऐसे ही बढ़ेंगे और T_{15} के पश्चात् फिर स्वत: ही T_0 उत्पन्न हो जायेगा। चित्र 1.6 में नीचे की तरंगे प्रदर्शित कर रही हैं, कि किस प्रकार $D_3T_4 = 1$ होने पर SC को क्लिअर (Clear) अर्थात् 0 कर दिया गया है। चित्र 1.6 में हम देख सकते हैं कि टाइमिंग सिगनल T_2 के पश्चात् ऑपरेशन डिकोडर का आउटपुट D_3 सक्रिय हो जाता है। जब टाइमिंग सिगनल T_4 सक्रिय होता है तो AND गेट के आउटपुट के तौर पर कण्ट्रोल फंक्शन D_3T_4 सक्रिय हो जाता है। इस सिगनल्स को SC के CLR पर प्रवाहित किया जाता है और अगले पॉजिटिव क्लॉक ट्रान्जिशन (Positive Clock Transition) पर काउन्टर को क्लिअर (Clear) अर्थात् 0 कर दिया जाता है। इसके कारण T_5 के स्थान T_0 टाइमिंग सिगनल उत्पन्न होता है।

अब हम कण्ट्रोल यूनिट (Control Unit) के आन्तरिक लॉजिक (Internal Logic) के विषय में पढ़ेंगे कि किस प्रकार कण्ट्रोल यूनिट (Control Unit) अपने इनपुट सिगनल (Input Signals) पर कार्यकर आउटपुट कण्ट्रोल (Output Control) को उत्पन्न करता है।

Scanned with CamScanner



चित्र **1.7** Micro Programmed Control

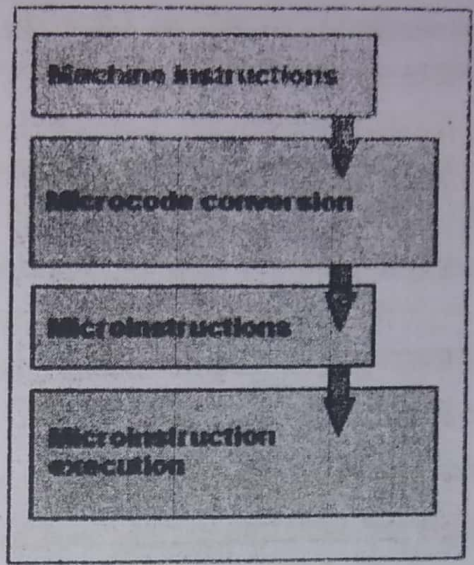


चित्र **1.8** Hard Wired Control

प्रश्न 17. CISC क्या है? CISC एवं RISC में तुलना कीजिए।

उत्तर CISC एक एकल निर्देश है। इसकी फुलफॉर्म Complex Instruction Set Computer है। ये assemb भाषा स्तर में बड़ी संख्या में अनेक निर्देशों की पूर्ति करता है। इसमें एक multiple निर्देश चक्र होता है और varia length निर्देश भी होते हैं।

CISC



CISC एवं RISC में तुलना निम्नलिखित है—

S.No.	RISC	CISC
1.	यह execution टाइम को reduce करता है।	Assembly भाषा स्तर में अधिक संख्या में कॉम्प्लैक्स निर्देशों की आपूर्ति करता है।
2.	इसमें एकल निर्देश चक्र होता है।	इसमें मल्टीपल निर्देश चक्र होता है।
3.	इसमें फिक्सड लैंथ निर्देश होते हैं।	इसमें वैरियेबल लैंथ निर्देश होते हैं।
4.	इसमें कुछ एड्रैसिंग मोड होते हैं।	इसमें मल्टीपल एड्रैसिंग मोड होते हैं।
5.	इसमें हार्डवायर्ड कण्ट्रोल होता है।	इसमें माइक्रोप्रोग्राम्ड कण्ट्रोल होता है।
6.	इसकी पाइप लाइनिंग काफी सरल होती है।	इसकी पाइप लाइनिंग कठिन होती है।
7.	इसमें रजिस्टरों का प्रयोग अधिक किया जाता है।	इसमें रजिस्टरों का प्रयोग कम किया जाता है।
8.	इसमें प्रति इंस्ट्रक्शन में 3 आपरेंड allow होते हैं।	इसमें केवल एक या दो आपरेंड allow होते हैं।

प्रश्न 18. RISC एवं CISC की विशेषताएँ लिखिए।

उत्तर RISC **एवं** CISC **की विशेषताएँ** RISC की विशेषताएँ निम्नलिखित हैं—

(i) इसमें सरल निर्देश डिकोडिंग होता है।
(ii) निर्देश एक शब्द के आकार के अन्तर्गत आते हैं।
(iii) निर्देश (Instruction) निष्पादित होने के लिए एकल घड़ी चक्र लेते हैं।
(iv) सामान्य प्रयोजन के लिए रजिस्टरों की संख्या अधिक होती है।
(v) सरल सम्बोधन मोड होते हैं।
(vi) इसमें डेटा कम लगता है।
(vii) इससे पाइपिंग प्राप्त की जा सकती है।

CISC की विशेषताएँ निम्नलिखित हैं—

(i) इससे जटिल निर्देश डिकोडिंग होती है।
(ii) निर्देश एक शब्द आकार से बड़े होते हैं।
(iii) निर्देश को निष्पादित करने के लिए एकल घड़ी चक्र से अधिक समय लगता है।
(iv) ऑपरेशन के दौरान सामान्य प्रयोजन रजिस्टर की कम संख्या स्मृति (memory) में ही प्रदर्शित हो जाती है।
(v) इसमें जटिल सम्बोधन मोड होता है।
(vi) अधिक डेटा की आवश्यकता होती है।

Scanned with CamScanner

2 मेमोरी ऑर्गेनाइजेशन
Memory Organisation

खण्ड 'अ' : अतिलघु उत्तरीय प्रश्न

प्रश्न 1. डिजिटल कम्प्यूटर में मेमोरी (Memory) क्यों आवश्यक है?

उत्तर किसी भी डिजिटल कम्प्यूटर में डेटा (Data) और प्रोग्राम्स (Programs) को सुरक्षित रखने के लिए मेमोरी की आवश्यकता पड़ती है।

प्रश्न 2. डिजिटल कम्प्यूटर में स्थिर मेमोरी प्रदान करने के लिए कितनी मेमोरी का उपयोग किया जाता है?

उत्तर डिजिटल कम्प्यूटर में स्थिर मेमोरी प्रदान करने के लिए पाँच प्रकार की मेमोरी का उपयोग किया जाता है।

प्रश्न 3. सीपीयू (CPU) कितने प्रकार की मेमोरी के सम्पर्क में रहता है?

उत्तर सीपीयू (CPU) केवल दो प्रकार की मेमोरी के सम्पर्क में रहता है।

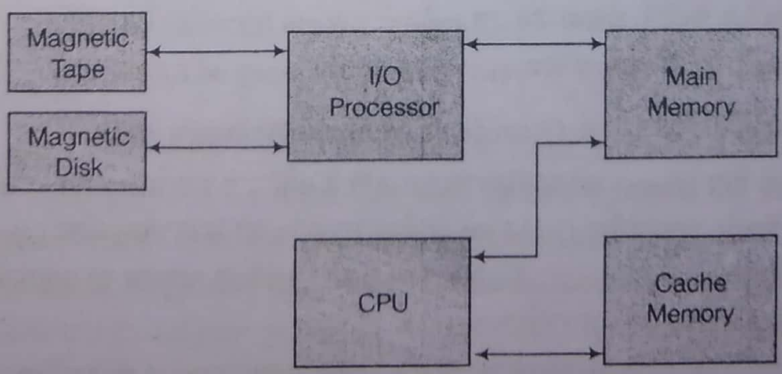
प्रश्न 4. डिवाइसेज में बैकअप को सुरक्षित रखने के लिए कौन-सी मेमोरी का उपयोग होता है?

उत्तर डिवाइसेज में बैकअप को सुरक्षित रखने के लिए ऑक्जिलरी मेमोरी (Auxiliary memory) का उपयोग होता है।

खण्ड 'ब' : लघु एवं दीर्घ उत्तरीय प्रश्न

प्रश्न 1. मेमोरी अनुक्रम (Memory Hierarchy) को चित्र सहित समझाइए।

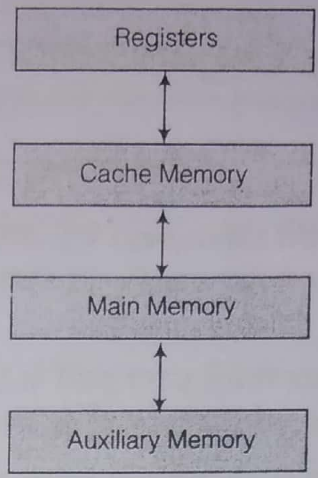
उत्तर किसी भी कम्प्यूटर में स्थित समस्त मेमोरीज (Memories) की क्षमता का हम मेमोरी अनुक्रम (Memory Hierarchy) द्वारा पता लगा सकते हैं। मेमोरी अनुक्रम (Memory Hierarchy) में कम्प्यूटर की समस्त मेमोरीज (Memories), धीमी परन्तु अधिक क्षमता वाली ऑक्जिलरी मेमोरी (Auxiliary Memory) से लेकर अधिक गति वाली मुख्य मेमोरी (Main Memory), से अत्यधिक गति वाले प्रोसेसिंग लॉजिक (Processing Logic) पर कार्य करने वाली कैशे मेमोरी (Cache Memory) सम्मिलित की जाती है। चित्र 2.1 में मेमोरी अनुक्रम (Memory Hierarchy) को प्रदर्शित किया गया है—



चित्र 2.1 Memory Hierarchy in a Computer System

उपरोक्त चित्र में दर्शायी गयी मेमोरी अनुक्रम (Memory Hierarchy) में हम देख सकते हैं कि ऑक्जिलरी मेमोरी (Auxiliary Memory), जिनकी गति कम और क्षमता अधिक होती है, से इनपुट-आउटपुट प्रोसेसर (I/O Processor) डेटा (Data) का लेन-देन करते हैं और इनपुट-आउटपुट प्रोसेसर (I/O Processor) से मेन मेमारी (Main Memory) में डेटा (Data) का लेन-देन (Transaction) किया जाता है, तत्पश्चात् या तो डेटा (Data) सीधे सीपीयू (CPU) में भेजा अथवा सीपीयू (CPU) से लिया जाता है या फिर कैशे मेमोरी (Cache Memory) का उपयोग किया जाता है। चित्र

2.1 में हमने कार्यप्रणाली के अनुसार मेमोरी अनुक्रम (Memory Hierarchy) का अध्ययन किया है। निम्नांकित चित्र 2.2 में हमने कीमत और श्रेष्ठता के बढ़ते क्रम में मेमोरी (Memory) को दर्शाया है।



चित्र 2.2

चित्र 2.2 में हम देख सकते हैं कि सबसे नीचे ऑक्जिलरी मेमोरी (Auxiliarly Memory) को दर्शाया गया है, क्योंकि यह मेमोरी, एक डिजिटल कम्प्यूटर में प्रयोग होने वाली समस्त मेमोरीज (Memories) में सबसे सस्ती होती है और इसकी प्रायोरिटी (Priority) भी सबसे कम होती है। इसके बाद मेन मेमोरी (Main Memory), जिसकी कीमत ऑक्जिलरी मेमोरी (Auxiliary Memory) से अधिक होती है। सबसे ज्यादा महँगे और सबसे ज्यादा प्रायोरिटी (Priority) वाले होते हैं—Registers, जो कि सीपीयू (CPU) के साथ सर्किट (Circuit) में ही बनाये गये होते हैं।

मेन मेमोरी (Main Memory)

किसी कम्प्यूटर में मेन मेमोरी (Main Memory) उस कम्प्यूटर की सेन्ट्रल स्टोरेज यूनिट (Central Storage Unit) होती है। कम्प्यूटर पर किये जाने वाले कार्यों के दौरान सीपीयू (CPU) इसी मेमोरी का प्रयोग करता है। यह मेमोरी पर्याप्त मात्रा में डेटा (Data) को सुरक्षित कर सकती है और इसकी कार्य गति भी बहुत अच्छी होती है। इस प्रकार की मेमोरीज (Memories) को बनाने के लिए सेमीकन्डक्टर इन्टीग्रेटेड सर्किट्स (Semiconductor Integrated Circuits) का प्रयोग किया जाता है। मेन मेमोरी (Main Memory) के अन्तर्गत मुख्यत: निम्नलिखित दो मेमोरीज (Memories) आती हैं—

(i) रैण्डम एक्सेस मेमोरी अथवा रैम (Random Access Memory or RAM)

(ii) रीड ओनली मेमोरी अथवा रोम (Read Only Memory or ROM)

रैण्डम एक्सेस मेमोरी अथवा रैम (Random Access Memory or RAM)

रैम (RAM) में डेटा (Data) को सुरक्षित किया जाता है और हम इस डेटा (Data) को रैण्डमली (Randomly) प्रयोग कर सकते हैं अर्थात् सम्पूर्ण रैम (RAM) में से डेटा (Data) को कैसे भी एक्सेस (Access) कर सकते हैं। रैम (RAM) इन्टीग्रेटेड सर्किट्स (Integrated Circuits—ICs) कार्य करने के अनुसार दो प्रकार में उपलब्ध हैं—स्टैटिक (Static) और डायनामिक (Dynamic)।

स्टैटिक रैम (Static RAM) में डेटा फ्लिप-फ्लाप (Flip-Flop) में सुरक्षित किया जाता है। फ्लिप-फ्लाप्स (Flip-Flops) में सुरक्षित डेटा (Data) तभी तक वैध (Valid) होता है, जब तक उनमें बिजली प्रवाहित की जाती है। इस प्रकार की रैम (RAM) में बिजली अधिक प्रयोग की जाती है।

डायनामिक रैम (Dynamic RAM) डेटा को इलैक्ट्रिक चार्ज (Electric Charges) के रूप में सुरक्षित रखता है जिन्हें कैपेसिटर्स (Capacitors) पर प्रवाहित किया जाता है। कैपेसिटर्स (Capacitors) पर प्रवाहित किया गया चार्ज (Charge) थोड़ी देर में स्वत: ही समाप्त किया जाता है, अत: डायनामिक रैम (Dynamic RAM) में प्रयोग किये गये

Scanned with CamScanner

कैपेसिटर्स (Capacitors) को लगातार रिचार्ज (Recharge) करना पड़ता है, और रिचार्ज (Recharge) करने के डायनामिक रैम (Dynamic RAM) को बार-बार रिफ्रेश (Refresh) किया जाता है। डायनामिक रैम (Dyna RAM) में बिजली की खपत कम होती है और ये एक मेमोरी चिप (Memory Chip) में अधिक डेटा को सुरक्षित की क्षमता रखता है।

स्टैटिक रैम (Static RAM) प्रयोग करने में अधिक सरल है और इसके रीड-राइट साइकल्स (Read-Write Cy भी काफी छोटे होते हैं। स्टैटिक रैम (Static RAM) की इन्हीं विशेषताओं के कारण इसका अत्यधिक उपयोग मेमोरी (Cache Memory) बनाने के लिए किया जाता है और डायनामिक रैम (Dynamic RAM) को सामान्य मेमोरी (Main Memory) बनाने के लिए किया जाता है। डायनामिक रैम (Dynamic RAM) के बेहतर (Versions) भी अब उपलब्ध हैं, जैसे कि Multibank DRAM, Extended dataout DRAM, Synchro DRAM और Direct RAM bus DRAM इत्यादि।

रीड ओनली मेमोरी अथवा रोम (Read Only Memory or ROM)

रीड ओनली मेमोरी अथवा रोम ROM (Read Only Memory or ROM) का प्रयोग उस डेटा (Data) को करने के लिए किया जाता है, जिसमें परिवर्तन नहीं किया जाना हो। कम्प्यूटर निर्माण के साथ ही रोम (ROM) का किया जाता है, इसमें कम्प्यूटर के वे सभी प्रोग्राम्स सुरक्षित किये जाते हैं जो कम्प्यूटर के सही प्रकार से कार्य क लिए आवश्यक हैं और ये प्रोग्राम्स अपरिवर्तनीय होते हैं। रोम (ROM) के अन्दर कॉन्सटेन्ट्स की तालिकायें (Tab Constants) भी सुरक्षित होती हैं।

इन सबके अतिरिक्त, रोम (ROM) की आवश्यकता एक अत्यन्त आवश्यक प्रोग्राम बूटस्ट्रैप लोडर (Boot Loader) को सुरक्षित करने के लिए होती है। बूटस्ट्रैप लोडर (Bootstrap Loader) एक ऐसा प्रोग्राम है, जिसक पॉवर (Power) ऑन (On) करने पर कम्प्यूटर को चालू करना होता है, बूटस्ट्रैप लोडर (Bootstrap Loade कम्प्यूटर को हमारे कार्य करने की स्थिति में लाता है। पॉवर (Power) ऑफ (Off) करने पर भी रोम (RO सुरक्षित डेटा (Data) पर कोई प्रभाव नहीं पड़ता। कम्प्यूटर को चालू करने में पॉवर (Power) ऑन (On) करन प्रारम्भिक प्रोग्राम्स (Initial Programs) को एग्जिक्यूट (Execute) करना होता है, अत: जब पॉवर (Power) को (On) किया जाता है, तब कम्प्यूटर हार्डवेयर, प्रोग्राम काउन्टर (Program Counter) में बूटस्ट्रैप लोडर (Boot Loader) का पहला एड्रैस (Address) डाल देता है। फिर बूटस्ट्रैप लोडर (Bootstrap Loader) डिस्क से ऑ सिस्टम (Operating System) की एक कॉपी मेन मेमोरी (Main Memory) में लाता है और नियन्त्रण ऑ सिस्टम को सौंप देता है। फिर ऑपरेटिंग सिस्टम कम्प्यूटर को सामान्य प्रयोग के लिए तैयार करता है।

रैम (RAM) और रोम (ROM) की चिप्स अनेक आकारों (Sizes) में उपलब्ध हैं। यदि आवश्यक मेमोरी एक चि क्षमता से अधिक है, तो हम कुछ चिप्स के संयोजन से आवश्यक मेमोरी को प्राप्त कर सकते हैं। यहाँ हम 12 RAM चिप्स और 512 × 8 ROM चिप्स की सहायता से 1024 × 8 मेमोरी बनायेंगे।

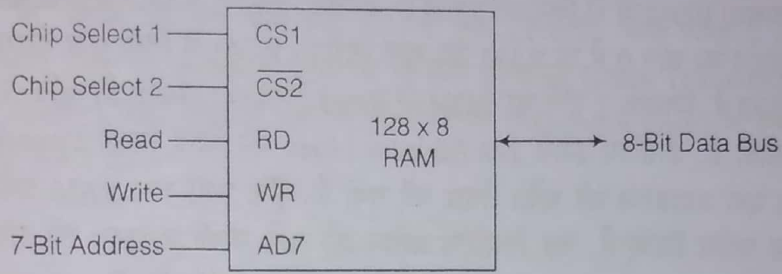
रैम और रोम चिप्स (RAM and ROM Chips)

रैम (RAM) चिप सीपीयू (CPU) के साथ संचार (Communication) के लिए तभी अच्छी होती हैं, जब उस अथवा अधिक कण्ट्रोल इनपुट्स (Control Inputs) होते हैं, जो चिप (Chip) को तभी सेलेक्ट करेंगे, जब आवश्यकता होगी। एक रैम (RAM) चिप का दूसरा गुण एक बाईडायरेक्शनल बस (Bidirectional Bus) है, द्वारा या तो रीड (Read) करते समय डेटा (Data) को मेमोरी (Memory) से सीपीयू (CPU) में भेजा जा सकता फिर राइट (Write) करते समय सीपीयू (CPU) से मेमोरी (Memory) में डेटा (Data) को भेजा जा सकता है बाईडायरेक्शनल डेटा बस (Bidirectional Data Bus) बनाने के लिए थ्री-स्टेट बफर (Three-State Buffe उपयोग किया जाता है।

Scanned with CamScanner

थ्री-स्टेट बफर (Three-State Buffer) का आशय एक ऐसे बफर (Buffer) से है, जिसकी तीन स्थितियाँ होती हैं और इसके आउटपुट में इन्हीं तीनों में से कोई एक स्थिति भेजी जाती है। तीन सम्भावित स्थितियाँ हैं—लॉजिक 1 के समतुल्य एक सिगनल, लॉजिक 0 के समतुल्य एक सिगनल और एक हाई-इम्पीडेन्स स्थिति (High-Impedance State)। सिगनल 1 और सिगनल 0, सामान्य सिगनल्स हैं, जबकि हाई-इम्पीडेन्स (High-Impedance) एक ओपन सर्किट (Open Circuit) की भाँति व्यवहार करता है, जिसका तात्पर्य है कि आउटपुट में कोई सिगनल नहीं है और न ही इसका कोई तार्किक अभिप्राय है।

चित्र 2.3 में 128 × 8 रैम (RAM) का ब्लॉक डायग्राम (Block Diagram) प्रदर्शित है। इस रैम (RAM) की क्षमता आठ बिट्स अर्थात् 1 बाइट के 128 वर्ड्स (Words) की है और इसमें 7-बिट्स के एड्रैस (Address) की ओर 8 बिट्स की बाईडायरेक्शनल डेटा बस (Bidirectional Data Bus) की आवश्यकता है।



चित्र **2.3** Block Diagram of RAM Chip

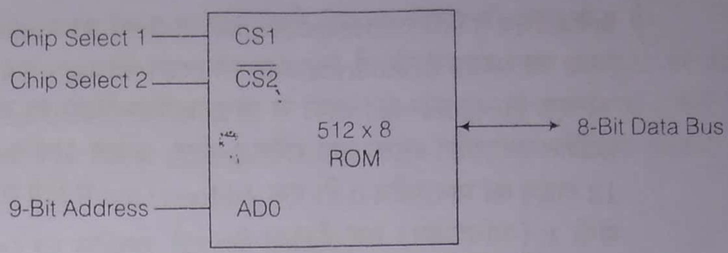
यहाँ रीड (Read) और राइट (Write) इनपुट, दोनों कार्यों को दर्शाया गया है। Chip Select अर्थात् *CS* नियन्त्रण के लिए प्रयोग किए गए इनपुट हैं, जोकि माइक्रोप्रोसेसर द्वारा चुने जाने पर चिप को सक्रिय करते हैं। एक से अधिक नियन्त्रण बिट्स (Control Bits) का प्रयोग तब अधिक सहायक होता है, जब कम्प्यूटर में एक से अधिक चिप होते हैं। जब चिप को चुना जाता है, तब उस पर रीड (Read) अथवा राइट (Write) के लिए सिगनल्स (Signals) को प्रवाहित किया जाता है।

RAM की कार्यप्रणाली। इसके लिए निम्नांकित तालिका का अध्ययन करें—

CS1	$\overline{CS2}$	RD	WR	Memory Function	State of Data Bus
0	0	×	×	Inhibit	High-Impedance
0	1	×	×	Inhibit	High-Impedance
1	0	0	0	Inhibit	High-Impedance
1	0	0	1	Write	Input Data to RAM
1	0	1	×	Read	Output Data from RAM
1	1	×	×	Inhibit	High-Impedance

उपरोक्त तालिका में हम देख सकते हैं कि जब CS1 और $\overline{CS2}$ का मान 0 है, तब RD और WR, डोन्ट केयर (Don't Care) स्थिति (Condition) में है अर्थात् इनके किसी भी मान में कोई असर नहीं पड़ेगा और मेमोरी (Memory) कोई फंक्शन नहीं करेगी और डेटा बस (Data Bus) हाई-इम्पीडेन्स (High-Impedence) की स्थिति में रहेगी। यदि हम ध्यान दें तो पायेंगे कि मेमोरी (Memory) केवल दो ही स्थितियों में कार्य करती है, पहली जब CS1 = 1 और $\overline{CS2}$ = 0 एवं Read = 0 और write = 1 है इस स्थिति में चूँकि Read = 0 है। इसलिए write फंक्शन किया जायेगा और बस (BUS) सीपीयू (CPU) से डेटा (DATA) लाकर रैम पर लिख देगी, और दसूरी जब CS = 1 और $\overline{CS2}$ = 0 तो है परन्तु Read = 1 है, इस स्थिति में Write डोन्ट केयर (Don't Care) स्थिति (Condition) में है। यदि Read = 1 है तो Write को देखे बिना Read फंक्शन किया जाता है और बस (Bus) रैम (RAM) से डेटा (Data) लेकर सीपीयू (CPU) तक जाती है।

Scanned with CamScanner



चित्र 2.4 Block Diagram of ROM Chip

अब हम रोम (ROM) चिप के विषय में चर्चा करेंगे। रोम (ROM) चिप बाहरी तौर पर रैम (RAM) चिप की ही भाँति संयोजित होती है। फिर भी चूंकि रोम (ROM) को केवल Read के लिए ही प्रयोग किया जा सकता है, इसलिए डेटा बस (Data Bus) को केवल आउटपुट के लिए ही प्रयोग किया जाता है। समान आकार की चिप में ROM में RAM से अधिक बिट्स आ सकती हैं क्योंकि ROM की आन्तरिक बायनरी Cells कम स्थान लेती है। 512 × 8 की ROM चिप का ब्लॉक डायग्राम (Block Diagram) पिछले चित्र में दर्शाया गया है।

यहाँ 9 बिट का एड्रैस (Address) प्रयोग किया जाता है। चिप को कार्य करने की स्थिति में लाने के लिए इनपुट CS1 = 1 और $\overline{CS2} = 0$ होना आवश्यक है, अन्यथा डेटा बस (Data Bus) हाई-इम्पीडेन्स (High-Impedence) की स्थिति में रहेगी। जब चिप को दोनों इनपुट कार्यकारी स्थिति में लाते हैं, तब 9 बिट के एड्रैस (Address) द्वारा चुनी गई बाइट को बस पर प्रवाहित किया जाता है।

प्रश्न 2. मेमोरी एड्रैस मैप (Memory Address Map) को तालिका सहित समझाकर लिखिए।

उत्तर **मेमोरी एड्रैस मैप** Memory Address Map कम्प्यूटर डिजाइनर को एक विशेष कार्य के लिए आवश्यक मेमोरी की मात्रा की गणना करनी चाहिए। आवश्यक मेमोरी का ज्ञान होने के पश्चात् RAM या ROM की उपलब्धता के आधार पर ही Memory और Processor के मध्य संपर्क बनाया जाता है। मेमोरी की एड्रैसिंग (Addressing) करने के लिए एक तालिका की सहायता ली जाती है, जिसके अन्दर प्रत्येक चिप को दिये गये एड्रैस (Address) लिखे होते हैं। इस तालिका को Memory Address Map कहते हैं।

उदाहरण के लिए, एक कम्प्यूटर लेते हैं जिसमें 512 बाइट्स की RAM और 512 बाइट्स की ROM की आवश्यकता है। RAM और ROM वर्गीकरण के लिए Memory Address Map निम्नांकित है—

तालिका 1

Components	**HexaDecimal**	**Address bus**									
	Address	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
RAM1	0000–007F	0	0	0	×	×	×	×	×	×	×
RAM2	0080–00FF	0	0	1	×	×	×	×	×	×	×
RAM3	0100–017F	0	1	0	×	×	×	×	×	×	×
RAM4	0180–01FF	0	1	1	×	×	×	×	×	×	×
ROM	0200–03FF	1	×	×	×	×	×	×	×	×	×

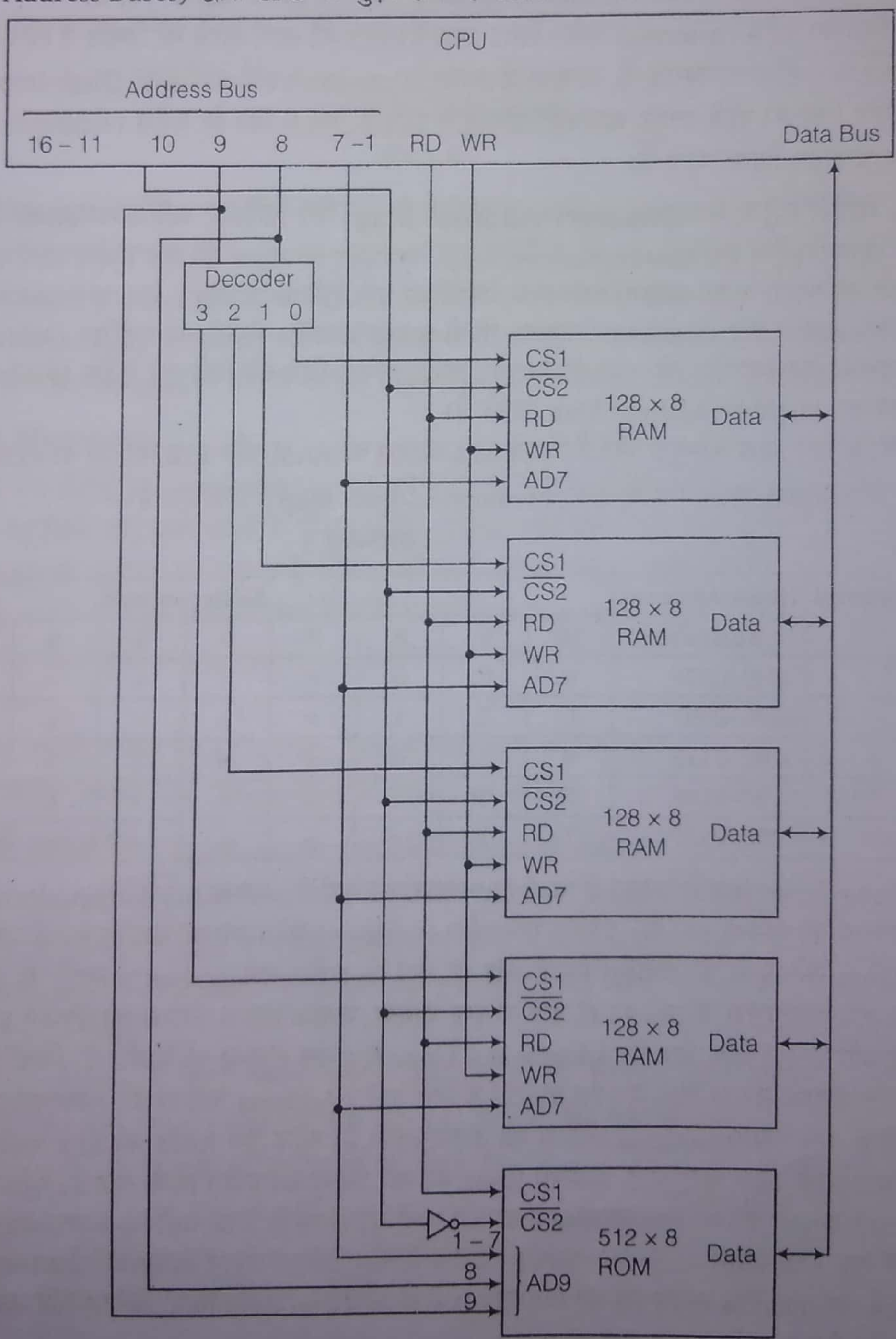
यहाँ Component कॉलम बताता है कि RAM प्रयोग की गयी है, अथवा ROM। Hexadecimal Address कॉलम में प्रत्येक चिप को एसाइन (Assign) किये गये एड्रैस (Address) की सीमा को दर्शाया गया है। तीसरे कॉलम में Address Bus Lines की सूची को प्रदर्शित किया गया है। वैसे तो एड्रैस बस (Address Bus) में 16 लाइन्स हैं, परन्तु इस उदाहरण में केवल 10 लाइन्स का ही प्रयोग किया जायेगा, जबकि शेष 6 लाइन्स का उपयोग इस उदाहरण में नहीं किया गया है और उसे 0 माना गया है। Address Bus Lines में × उन लाइन्स को दर्शाता है, जिन्हें प्रत्येक चिप के Address में जोड़ना पड़ेगा। RAM चिप में 128 बाइट्स हैं और उसे 7 Address लाइन्स की आवश्यकता है। ROM चिप में 512 बाइट्स हैं और उसे 9 Address लाइन्स की आवश्यकता है। चूँकि हमें RAM की 512 बाइट्स चाहिए और एक चिप की क्षमता सिर्फ 128 बाइट्स है, इसलिये RAM की चार चिप्स का प्रयोग किया गया है। चूँकि 128 बाइट की एक चिप के लिए 7 लाइन्स की आवश्यकता है इसलिए इस कार्य को करने के लिए चारों को अलग-अलग Address प्रदान किया गया है, इन उदाहरण में Address लाइन 8 और 9 में चार अलग-अलग बायनरी संयोजन बनाये गये हैं। यहाँ हम देख सकते हैं कि Address लाइन 10 का मान यदि 0 है तो RAM का प्रयोग किया जायेगा और यदि यह मान 1 है तो ROM का प्रयोग किया जायेगा।

Scanned with CamScanner

प्रत्येक चिप में दिये गये Hexadecimal Address को Address Bus Assignment में दी गयी सूचना के अनुसार प्राप्त किया जा सकता है। Address Bus की लाइन को चार-चार के समूह में विभाजित किया गया है, जिससे प्रत्येक समूह को एक Hexadecimal अंक के द्वारा प्रदर्शित किया जा सकता है। पहला Hexadecimal अंक 13–16 समूह को प्रदर्शित करता है जिसका मान सदैव 0 होगा, अगला अंक 9–12 समूह को प्रदर्शित करता है, परन्तु इस समूह 11 और 12 लाइन का मान सदैव 0 है। एक Address Line में दिये गये × का तात्पर्य है कि उसकी सीमा समस्त 0 (न्यूनतम) से सभी 1 (अधिकतम) तक है।

प्रश्न 3. सीपीयू (CPU) में मेमोरी कनेक्शन को चित्र सहित समझाइए।

उत्तर **सीपीयू से मेमोरी का कनेक्शन** Memory Connection to CPU RAM और ROM डेटा एवं एड्रैस बसेज (Data and Address Buses) द्वारा CPU से जुड़ी होती है। Address Bus की निचले स्तर की लाइन Chips में से



चित्र **2.5 Memory connection to CPU**

Scanned with CamScanner

Access की जाने वाली बाइट को चुनती हैं और शेष लाइन्स Chip Select इनपुट द्वारा किसी विशेष चिप को चुन
CPU से मेमोरी के Connection को चित्र 2.5 में दर्शाया गया है।
चित्र 2.5 में 512 MB RAM और 512 MB ROM को दर्शाया गया है। इसके अन्दर प्रश्न 2 में दी गयी तालिक
दर्शाये गये Memory Map का प्रयोग किया गया है। प्रत्येक RAM अपनी 128 सम्भावित बाइट्स में से चुनने के
Address Bus निचले स्तर की सात बिट्स को प्राप्त करती है। किस चिप को चुनना है, यह निर्धारण Address B
8 और 9 लाइन्स के द्वारा किया जाता है। इस कार्य को करने के लिए 2 × 4 Decoder का प्रयोग किया जाता है,
आउटपुट प्रत्येक RAM चिप में CS1 इनपुट की भाँति प्रवाहित किया जाता है। अत: जब 8 व 9 लाइन 00
RAM1 को चुना जायेगा, जब यह 01 है तो RAM2, 10 है तो RAM3 व 11 है तो RAM4 को चुना जायेगा।
RD और WR आउटपुट को प्रत्येक RAM चिप में इनपुट की भाँति प्रवाहित किया जाता है।
अब, RAM अथवा ROM में से किसको चुनना है, इसका निर्धारण Address Bus की 10 लाइन के अनुसार कि
है। यदि 10 लाइन का मान 0 है तो RAM को चुना जायेगा। ROM में दिया गया दूसरा Chip Select CPU के
RD से जुड़ा हुआ है, जिसके 1 होने पर ROM से Read किया जा सकता है। चूँकि ROM को 9-बिट के Addr
आवश्यकता होती है, इसलिए इसमें 1-9 Address Lines को बिना किसी Decoder के प्रवाहित किया गया
यह चित्र 2.5 एक उदाहरण की भाँति तैयार की गयी है चूँकि यहाँ चार RAM प्रयोग की गयी हैं, इसलिए 2
Decoder का प्रयोग किया है, यह निर्धारण प्रयोग की जाने वाली RAM's की संख्या पर निर्भर करता है।

प्रश्न 4. सहायक मेमोरी (Auxiliary Memory) की कार्यप्रणाली को समझाइए।

उत्तर **सहायक मेमोरी** Auxiliary Memory कम्प्यूटर सिस्टम में मुख्यत: प्रयोग की जाने वाली सहायक (Auxiliary Memory) मैग्नेटिक डिस्क (Magnetic Disk) और मैग्नेटिक टेप (Magnetic Tape) हैं। मै ड्रम्स (Magnetic Drums), मैग्नेटिक बबल मेमोरी (Magnetic Bubble Memory) और ऑप्टिकल (Optical Disk) भी सहायक मेमोरी (Auxiliary Memory) के उदाहरण हैं, परन्तु इन्हें इतना अधिक प्रयोग नह जाता।

सहायक मेमोरी (Auxiliary Memory) की कार्यप्रणाली को समझने के लिए, मैग्नेटिक्स (Magnetics), इलेक्ट्र (Electronics) और इलेक्ट्रोमैकेनिकल (Electromechanical) सिस्टम (System) का ज्ञान होना आवश्य यद्यपि इन डिवाइसेज (Devices) के भौतिक गुण थोड़े जटिल (Complex) हैं, परन्तु इनके लॉजिक (Logic) गु विशेषताओं और कुछ मानकों द्वारा इनकी तुलना की जा सकती है।

किसी डिवाइस (Device) के मुख्य गुण एक्सेस मोड (Access Mode), एक्सेस टाइम (Acess Time), ट्रान्स (Transfer Rate), क्षमता (Capacity) और लागत (Cost) हैं।

1. **एक्सेस मोड** Access Mode कम्प्यूटर में स्थित डेटा (Data) को किस प्रकार प्रयोग किया जा सकता एक्सेस मोड (Access Mode) कहा जाता है। यह दो प्रकार का होता है—सीक्वेंशियल (Sequentia रैण्डम (Random)। सीक्वेंशियल एक्सेस मोड (Sequential Access Mode) में हम डिवाइस (Devi प्रारम्भ से एक्सेस (Access) करना प्रारम्भ करेंगे और रैण्डम एक्सेस मोड (Random Access Mode) डेटा (Data) को पूरी डिवाइस (Device) में से कहीं से भी एक्सेस (Access) करते हैं।
2. **एक्सेस टाइम** Access Time मेमोरी में किसी विशेष स्टोरेज लोकेशन (Storage Location) को ढूं उसके कन्टेन्ट्स (Contents) को एक्सेस (Access) करने में लगने वाले औसत समय (Average Tim एक्सेस टाइम (Access Time) कहा जाता है। एक्सेस टाइम (Access Time) में सीक टाइम (Seek अर्थात् एक लोकेशन (Location) पर Read-Write को लाने में लगने वाला समय और ट्रान्सफर (Transfer Time) अर्थात् एक डिवाइस (Device) से डेटा ट्रान्सफर (Data Transfer) करने में लग समय सम्मिलित होता है।
3. **ट्रान्सफर रेट** Transfer Rate एक डिवाइस (Device) से किसी अन्य डिवाइस (Device) में डेटा (D जाने में लगने वाले समय को ट्रान्सफर टाइम (Transfer Time) और प्रति सेकण्ड किये जाने वाले (Transfer) को ट्रान्सफर रेट (Transfer Rate) कहा जाता है।

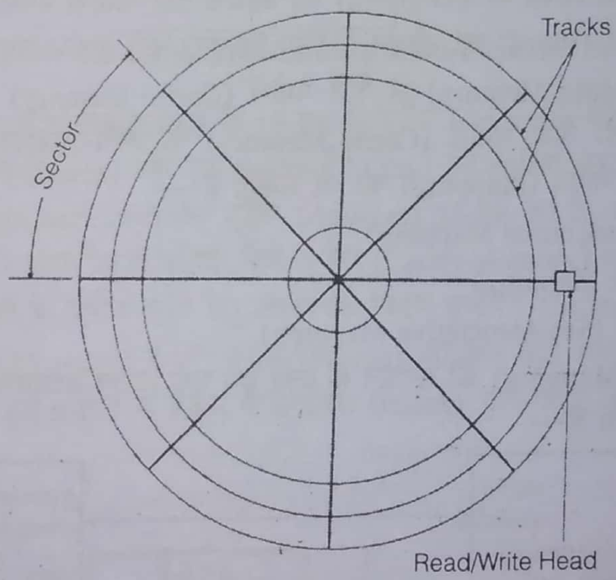
4. **कैपेसिटी** Capacity एक डिवाइस (Device) में सुरक्षित किये जा सकने वाले कुल डेटा (Data) को उस डिवाइस (Device) की क्षमता अर्थात् कैपेसिटी (Capacity) कहा जाता है।

5. **कॉस्ट** Cost एक डिवाइस (Device) का मूल्य भी डिवाइस (Device) का चुनाव करने में सहायता करता है।

प्रश्न 5. मैग्नेटिक डिस्क (Magnetic Disks) क्या है? इसको चित्र सहित समझाइए।

उत्तर **मैग्नेटिक डिस्क** Magnetic Disks मैग्नेटिक डिस्क, मेटल (Metal) अथवा प्लास्टिक (Plastic) की एक गोल प्लेट होती है, जिस पर मैग्नेटाइज्ड मैटेरियल (Magnetized Material) की कोटिंग (Coating) की होती है। अधिकांश डिस्क की दोनों साइड्स (Sides) का उपयोग किया जाता है और एक धुरी (Spindle) पर कुछ डिस्क को इकट्ठा किया जाता है तथा प्रत्येक सतह पर रीड/राइट हैड (Read/Write Head) उपलब्ध होता है।

सभी डिस्क्स एक साथ बहुत ज्यादा गति से घूमती हैं और एक्सेस (Access) करने के लिए ये ना तो रुकती हैं और ना ही चलती हैं। एक डिस्क पर अनेक वृत्ताकार भाग होते हैं, इन्हें ट्रैक (Track) कहा जाता है और ट्रैक को सेक्टर्स (Sectors) में विभाजित किया जाता है। अधिकांश सिस्टम्स में डेटा ट्रान्सफर (Data Transfer) की जा सकने वाली न्यूनतम इकाई को सेक्टर (Sector) कहा जाता है। एक मैग्नेटिक डिस्क (Magnetic Disk) के इन समस्त भागों को हम निम्नांकित चित्र द्वारा समझ सकते हैं—



चित्र **2.6** Magnetic Disk

कुछ यूनिट्स (Units) में प्रत्येक डिस्क सरफेस (Disk Surface) के लिए एक ही रीड/राइट हैड (Read/Write Head) होता है। इस प्रकार की डिवाइस (Device) में रीड अथवा राइट (Read or Write) करने हेतु मैकेनिकल एसेम्बली (Mechanical Assembly) द्वारा ट्रैक एड्रैस बिट्स (Track Address Bits) का प्रयोग हैड (Head) को वांछित स्थिति पर पहुँचाने के लिए किया जाता है।

दूसरे प्रकार की डिवाइसों (Devices) में प्रत्येक ट्रैक (Track) के लिए पृथक् रीड/राइट हैड (Read/Write Head) उपलब्ध होता है। यहाँ एड्रैस बिट डिकोडर (Address Bit Decoder) के द्वारा एक विशेष ट्रैक (Track) को इलैक्ट्रॉनिकली (Electronically) चुन लिया जाता है। इस प्रकार की डिवाइसेज (Devices) ज्यादा महँगी होती हैं, और बहुत बड़े कम्प्यूटर सिस्टम में पायी जाती हैं।

एक डिस्क सिस्टम के एड्रैस (Address) में डिस्क नम्बर (Disk Number), डिस्क सरफेस (Disk Surface), सेक्टर नम्बर (Sector Number) और सेक्टर में ट्रैक्स (Tracks in Sector) सम्मिलित होते हैं। निर्धारित ट्रैक (Track) पर रीड/राइट हैड के पहुँचने के पश्चात् सिस्टम को तब तक प्रतीक्षा करनी पड़ती है, जब तक घूमती हुई डिस्क में रीड/राइट हैड (Read/Write Head) वांछित सेक्टर (Sector) पर नहीं पहुँच जाता। एक बार सेक्टर की शुरुआत पर पहुँचने के पश्चात् डेटा ट्रान्सफर (Data Transfer) तीव्र गति से हो जाता है।

Scanned with CamScanner

परिधि के पास वाले ट्रैक का सेक्टर (Sector) केन्द्र बिन्दु के पास वाले ट्रैक के सेक्टर से बड़ा होता है।

वे डिस्क्स (Disks) जो कम्प्यूटर में स्थाई (Permanent) रूप से जुड़ी होती हैं और जिनके बिना कम्प्यूटर पर कार्य नहीं किया जा सकता, उन्हें हार्ड डिस्क (Hard Disk) कहा जाता है। वे डिस्क जो कम्प्यूटर से हटायी जा सकती है, उन्हें फ्लॉपी डिस्क (Floppy Disk) और कॉम्पेक्ट डिस्क (Compact Disk) कहा जाता है। ये विभिन्न आकारों में उपलब्ध हैं।

प्रश्न 6. मैग्नेटिक टेप (Magnetic Tape) पर संक्षिप्त टिप्पणी लिखिए।

उत्तर **मैग्नेटिक टेप** Magnetic Tape मैग्नेटिक टेप (Magnetic Tape) द्वारा डेटा ट्रांसफर (Data Transfer) में इलेक्ट्रिकल (Electrical), मैकेनिकल (Mechanical) और इलेक्ट्रॉनिक (Electronic) तत्त्व सम्मिलित होते हैं, जो मैग्नेटिक टेप यूनिट के विभिन्न भाग और उन पर नियन्त्रण करने के माध्यम उपलब्ध कराते हैं। मैग्नेटिक टेप (Magnetic Tape), मैग्नेटिक रिकॉर्डिंग मीडियम (Magnetic Recording Medium) के द्वारा कोटेड (Coated) प्लास्टिक (Plastic) की एक पट्टी है।

डेटा बिट्स को मैग्नेटिक टेप (Magnetic Tape) पर मैग्नेटिक स्पॉट्स (Magnetic Spots) के रूप में विभिन्न ट्रैक्स (Tracks) पर रिकॉर्ड किया जाता है। सामान्यत: पैरिटी बिट (Parity Bit) के साथ एक कैरेक्टर (Character) को बनाने के लिए 7 अथवा 9 बिट्स को एक साथ रिकार्ड करना होता है। डेटा को सिक्वेन्स (Sequence) में रिकार्ड (Record) और रीड (Read) करने के लिए प्रत्येक ट्रैक (Track) पर रीड/राइट हैड (Read/Write Head) बनाया जाता है।

मैग्नेटिक टेप को रोकना, फॉरवर्ड (Forward) करना, रिवर्स (Reverse) करना और रि-राइट (Re-write) करना सम्भव है। इस टेप के शुरू और अन्त में एक विशेष बिट पैटर्न (Bit Pattern) दिया होता है, जिससे हमें पता चलता है कि टेप का आदि और अन्त कहाँ है।

एक टेप के एड्रैस (Address) में रिकॉर्ड नम्बर (Record Number) और रिकॉर्ड में कैरेक्टर्स की संख्या (Number of Characters in Record) सम्मिलित होती हैं। रिकॉर्ड (Record) का आकार निश्चित (Fixed) अथवा परिवर्तनीय (Variable) हो सकता है।

प्रश्न 7. एसोसिएटिव मेमोरी (Associative Memory) की कार्यविधि का वर्णन कीजिए।

उत्तर **एसोसिएटिव मेमोरी** Associative Memory अनेक डेटा प्रोसेसिंग एप्लीकेशन्स (Data Processing Applications) में मेमोरी में सुरक्षित तालिका के आइटम्स (Items) को ढूंढने की आवश्यकता होती है। उदाहरण के लिए; एक बैंक में एकाउण्ट धारक (Account Holder) का नाम और एकाउण्ट स्टेटस (Account Status) को जानने के लिए एक फाइल में सुरक्षित एकाउण्ट नम्बर्स (Account Numbers), में उसका एकाउण्ट नम्बर खोजना होता है। किसी भी तालिका में से किसी आइटम को खोजने के लिए समस्त आइटम्स की एक श्रेणी (Sequence) में सुरक्षित होना आवश्यक समझा गया है। किसी भी आइटम को खोजने के लिए सर्वप्रथम एड्रैसेज (Addreses) की श्रेणी (Sequence) को चुना जाता है, तत्पश्चात् प्रत्येक एड्रैस (Address) पर सुरक्षित सूचना को खोजे जाने वाले आइटम से तुलना तब तक की जाती है जब तक वांछित आइटम नहीं मिल जाता। हमें कितनी बार तुलना करनी पड़ी, यह खोजे जाने वाले आइटम और प्रयुक्त की गयी एल्गोरिथ्म (Algorithm) पर निर्भर करता है।

यदि मेमोरी में सुरक्षित डेटा (Data) को एड्रैस (Address) के स्थान पर डेटा (Data) के कन्टेन्ट (Content) से ही पहचाना जाये, तो मेमोरी में सुरक्षित आइटम (Item) को खोजने में लगने वाले समय को कम किया जा सकता है। वह मेमोरी (Memory), जिसे कन्टेन्ट (Content) के द्वारा प्रयोग किया जाता है, एसोसिएटिव मेमोरी (Associative Memory) अथवा कन्टेन्ट एड्रैसेबल मेमोरी (Content Addressable Memory—CAM) कहा जाता है।

इस प्रकार की मेमोरी में डेटा (Data) को एड्रैस (Address) के स्थान पर कन्टेन्ट (Content) की सहायता से खोजा जाता हैं जब किसी वर्ड (Word) को एसोसिएटिव मेमोरी (Associative Memory) में सुरक्षित किया जाता है, तब उसे कोई एड्रैस (Address) प्रदान नहीं किया जाता। इस प्रकार की मेमोरी किसी वर्ड (Word) को सुरक्षित करने के लिए अनयूज्ड (Unused) स्थान को खोजने में स्वयं सक्षम होती है। जब भी एसोसिएटिव मेमोरी (Associative Memory)

में से किसी वर्ड (Word) को खोजना होता है, तब उसका कोई भाग अथवा हिस्सा बता दिया जाता है, तत्पश्चात् मेमोरी उन सभी वड्र्स (Words) को रीडिंग (Reading) के लिए चिह्नित कर देती है, जिनमें बताया गया भाग सम्मिलित होता है।

एसोसिएटिव मेमोरी (Associative Memory) को विशेषत: समानान्तर खोजों (Parallel Searches) के लिए प्रयोग किया जाता है। एसोसिएटिव मेमारी (Associative Memory), रैण्डम एक्सेस मेमोरी (Random Access Memory) से अधिक महँगी होती है, क्योंकि इसके प्रत्येक सेल (Cell) में स्टोरेज क्षमता (Storage Capacity) के साथ-साथ तुलना करने के लिए आवश्यक लॉजिक (Logic) का होना भी आवश्यक है। इसी कारण से एसोसिएटिव मेमोरी (Associative Memory) का उपयोग वहाँ किया जाता है, जहाँ सर्च टाइम (Search Time) का बहुत छोटा होना और अत्यन्त नाजुक (Critical) होना आवश्यक होता है।

प्रश्न 8. कैशे मेमारी (Cache Memory) का वर्णन कीजिए।

उत्तर **कैशे मेमोरी** Cache Memory अनेक विशिष्ट प्रोग्राम्स का विश्लेषण (Analysis) करने पर देखा गया है, कि एक विशेष समय अन्तराल में मेमोरी (Memory) को दिये गये रेफरेन्सेज (References) मेमोरी के कुछ निर्दिष्ट क्षेत्र तक ही सीमित हो जाते हैं। इस तथ्य को लोकेलिटी ऑफ रेफरेन्स (Locality of Reference) की प्रॉपर्टी (Property) के रूप में जाना जाता है। जब प्रोग्राम्स में एक सबरूटीन (Subroutine) को एग्जिक्यूट (Execute) किया जाता है, तो इसमें सुरक्षित निर्देशों (Instructions) को मेमोरी में लाया जाता है। जब प्रोग्राम्स में एक लूप (Loop) को एग्जिक्यूट (Execute) किया जाता है। तो लूप (Loop) में दिए गए निर्देशों (Instructions) को मेमोरी में से पुनरावृत्ति में प्रयोग किया जाता है। अत: निर्देशों (Instructions) के फैच (Fetch) करने में लूप (Loop) और सबरूटीन्स (Subroutines) में लोकेलिटी ऑफ रेफरेन्स (Locality of Reference) का प्रयोग होता है।

यदि प्रोग्राम्स और डेटा के सक्रिय भागों को एक तीव्र और छोटी मेमोरी में रख दिया जाये तो, औसत एक्सेस टाइम (Average Access Time) को कम किया जा सकता है और इस प्रकार प्रोग्राम्स के कुल एग्जिक्यूशन टाइम (Execution Time) को कम किया जा सकता है। इस प्रकार की छोटी और तीव्र मेमोरी को कैशे मेमोरी (Cache Memory) कहा जाता है। इसे दिये गये चित्र की भाँति सीपीयू (CPU) और मेन मेमोरी (Main Memory) के मध्य लगाया जाता है। कैशे मेमोरी (Cache Memory) का एक्सेस टाइम (Access Time), मेन मेमोरी (Main Memory) के एक्सेस टाइम (Access Time) का 5 से 10वां भाग होता है। मेमोरी अनुक्रम (Memory Hierarchy) में कैशे (Cache) की गति सबसे अधिक होती है और लगभग सीपीयू (CPU) के अवयव (Components) के समान होती है।

कैशे मेमोरी (Cache Memory) में सर्वाधिक प्रयोग होने वाले निर्देश और डेटा को रखा जाता है, जिससे औसत मेमोरी एक्सेस टाइम (Average Memory Access Time) लगभग कैशे मेमोरी (Cache Memory) के एक्सेस टाइम (Access Time) के समान हो जाता है। कैशे मेमोरी (Cache Memory) का आकार मेन मेमोरी (Main Memory) के आकार का एक छोटा-सा भाग है, इसलिए इसमें सभी निर्देशों (Instructions) को रखना असम्भव है और इसीलिए इसमें उन निर्देशों (Instructions) को रखा जाता है जो लोकेलिटी ऑफ रेफरेन्स (Locality of Reference) पर कार्य करती है।

कैशे (Cache) की कार्यप्रणाली के अनुसार जब सीपीयू (CPU) को मेमोरी (Memory) को एक्सेस करने की आवश्यकता होती है, तो पहले कैशे (Cache) को जांचा जाता है। यदि वांछित वर्ड (Word), कैशे (Cache) में प्राप्त हो जाता है, तो उसे रीड (Read) कर लिया जाता है और यदि सीपीयू (CPU) से एड्रैस (Address) किया गया वर्ड (Word), कैशे (Cache) में प्राप्त नहीं होता है, तो उस वर्ड (Word) को रीड (Read) करने के लिए मेन मेमोरी (Main Memory) का प्रयोग किया जाता है। वड्र्स (Words) के एक ब्लॉक (Block) को, जिसमें वांछित वर्ड (Word) भी होता है, को मेन मेमोरी (Main Memory)से कैशे मेमोरी (Cache Memory) में भेज दिया जाता है। इस प्रकार कुछ डेटा (Data) को कैशे मेमोरी (Cache Memory) में स्थानान्तरित कर दिया जाता है, ताकि मेमोरी को भविष्य में होने वाले रेफरेन्सेज (References) के वड्र्स (Words), कैशे मेमोरी (Cache Memory) में ही मिल जायें।

Scanned with CamScanner

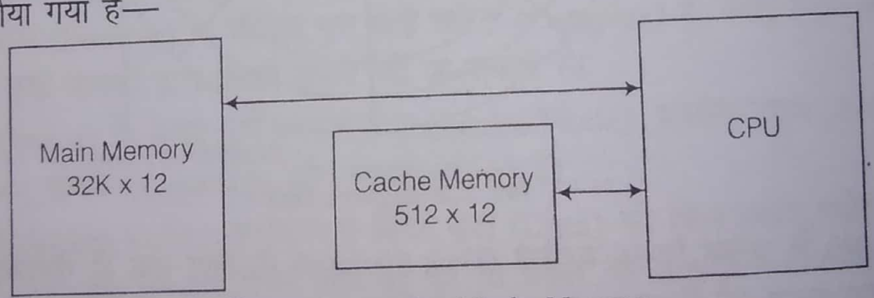
जब सीपीयू (CPU) मेमोरी (Memory) को रेफर (Refer) करता है और वर्ड (Word) कैशे (Cache) में प्राप्त है जाता है जो इसे हिट (Hit) उत्पन्न करना, और यदि वर्ड (Word) कैशे (Cache) में प्राप्त नहीं होता है तो, इसे मि (Miss) उत्पन्न करना कहा जाता है। कैशे मेमोरी (Cache Memory) की परफॉरमेन्स (Performance) की गणन (Hit Ratio) के द्वारा की जाती है। इसकी गणना करने के लिए कुल हिट्स (Hits) की संख्या में और सीपीयू (CPU द्वारा किये गये कुल मेमोरी रेफरेन्सेज (Memory References), जिसमें समस्त हिट्स (Hits) और मिसेस (Misses सम्मिलित होते हैं, से भाग किया जाता है। हिट रेशियो (Hit Ratio) को 0.9 अथवा उससे ज्यादा भी देखा गया है, ज लोकेलिटी ऑफ रेफरेन्स (Locality of Reference) के सिद्धान्त को सिद्ध करता है।

कैशे (Cache) के प्रयोग द्वारा औसत मेमोरी एक्सेस टाइम (Average Memory Access Time) को बढ़ाया जा सक है। यदि हिट रेशियो (Hit Ratio) का मान अत्यधिक है, तो सीपीयू (CPU) अधिकांशतः डेटा को मेन मेमोरी (Ma Memory) के स्थान पर कैशे मेमोरी (Cache Memory) से प्राप्त करता है, जिससे औसत मेमोरी एक्सेस टा (Average Memory Access Time), कैशे (Cache) के तीव्रतर एक्सेस टाइम (Faster Access Time) के लगभ बराबर हो जाता है।

कैशे (Cache) का मुख्य गुण किसी भी डेटा (Data) को अधिक तीव्र गति से एक्सेस (Access) करना है। इसीलि कैशे (Cache) में किसी वर्ड (Word) को खोजने के लिए बिल्कुल नहीं अथवा बहुत ही थोड़ा समय लगता है। डे (Data) को मेन मेमोरी (Main Memory) से कैशे मेमोरी (Cache Memory) में भेजने की प्रक्रिया को मैपि (Mapping) कहा जाता है। कैशे मेमोरी (Cache Memory) के ऑर्गेनाइजेशन (Organisation) के अनुस निम्नलिखित तीन प्रकार से मैपिंग (Mapping) की जा सकती है—

1. ऐसोसिएटिव मैपिंग (Associative Mapping)
2. डायरेक्ट मैपिंग (Direct Mapping)
3. सेट-ऐसोसिएटिव मैपिंग (Set-Associative Mapping)

इन तीनों प्रकार की मैपिंग (Mapping) को समझने के लिए हम एक विशेष उदाहरण की सहायता लेंगे। इस उदाहर को चित्र 2.7 में दर्शाया गया है—



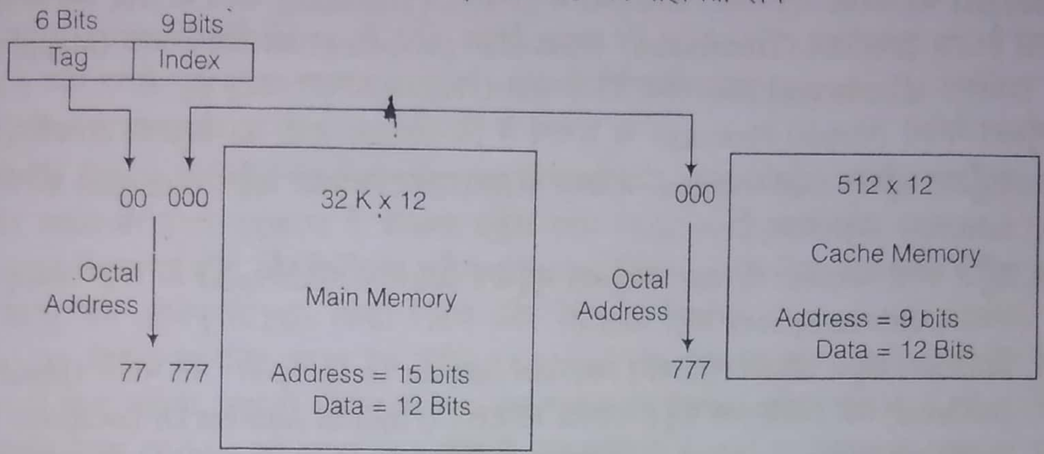
चित्र 2.7 Example of Cache Memory

यहाँ मेन मेमोरी (Main Memory), 12 बिट्स के 32K वर्ड्स को स्टोर (Store) कर सकती है। कैशे (Cache) दिये गये किसी एक समय पर वर्ड्स (Words) के 512 वर्ड्स को स्टोर (Store) किया जा सकता है। कैशे (Cach में स्टोर किये गये प्रत्येक वर्ड (Word) की एक कॉपी (Copy) मेन मेमोरी (Main Memory) में स्टोर (Store) है है। सीपीयू (CPU) दोनों मेमोरीज (Memories) से कम्प्यूनिकेट (Communicate) कर सकता है। सीपीयू (CP पहले 15-Bit का एक एड्रैस (Address) कैशे (Cache) को भेजता है, यदि डेटा (Data) प्राप्त हो जाता है, अर्थात् ि (Hit) है तो, सीपीयू (CPU) कैश (Cache) से 12-Bit का डेटा स्वीकार करता है। यदि डेटा (Data) कैशे (Cach में नहीं मिलता है, अर्थात् मिस (Miss) है, तो सीपीयू (CPU) डेटा (Data) को मेन मेमोरी (Main Memory) से (Read) करता है और तत्पश्चात् वर्ड (Word) को कैशे (Cache) में भेज दिया जाता है।

Scanned with CamScanner

प्रश्न 9. डायरेक्ट मैपिंग को समझाइए।

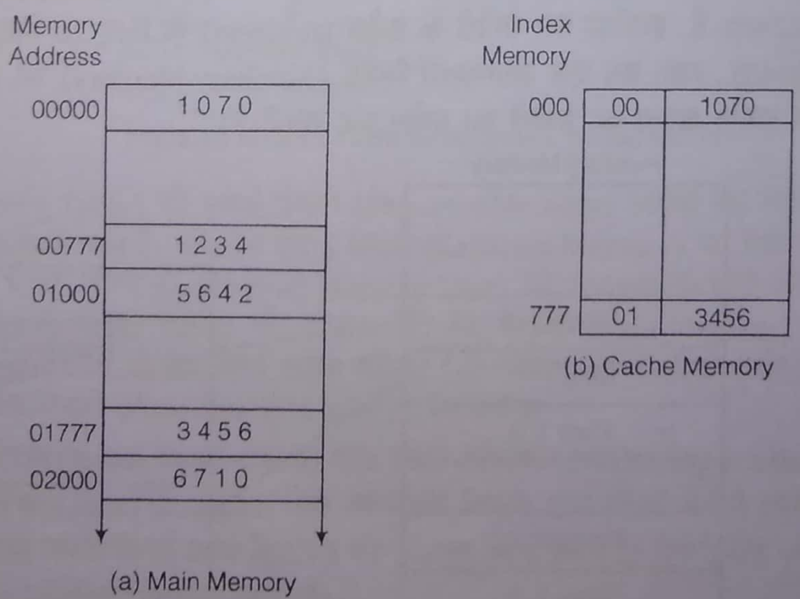
उत्तर **डायरेक्ट मैपिंग** Direct Mapping प्रत्येक सेल (Cell) में जोड़े गये लॉजिक (Logic) के कारण एसोसिएटिव मेमोरी (Associative Memory) का मूल्य रेन्डम-एक्सेस मेमोरी (Random-Access Memory) के मूल्य से अपेक्षाकृत अधिक होता है। हम रेन्डम-एक्सेस मेमोरी (Random-Access Memory) को भी कैशे (Cache) के लिए प्रयोग कर सकते हैं। इस प्रयोग को समझने के लिए चित्र 2.8 की सहायता लेंगे—



चित्र **2.8** Addressing Relationship between Main and Cache Memory

इस मैपिंग (Mapping) में सीपीयू (CPU) के 15-Bit के एड्रैस (Address) को दो भागों में विभाजित किया जाता है। अन्तिम 9 बिट्स इन्डैक्स (Index) और शेष 6-Bits टैग (Tag) को प्रदर्शित करती है। उपरोक्त चित्र में हम देख सकते हैं कि मेन मेमोरी (Main Memory) का एड्रैस (Address) 15-Bit का है, अत: इसमें टैग (Tag) और इन्डैक्स (Index) दोनों भागों की आवश्यकता पड़ेगी। कैशे मेमोरी (Cache Memory) का एड्रैस 9-Bit का है, अत: हम कह सकते हैं कि कैशे मेमोरी के प्रयोग करने हेतु आवश्यक बिट्स इन्डैक्स (Index) भाग में दी गयी है।

यदि मेन मेमोरी (Main Memory) में 2^n वड्र्स (Words) को और कैशे मेमोरी में 2^k वड्र्स (Words) को स्टोर किया जा सकता है, तो हम n बिट के एड्रैस के इन्डैक्स (Index) भाग को k बिट्स का और टैग (Tag) को $n-k$ बिट्स का बनायेंगे।



चित्र **2.9** Direct Mapping Cache Organisation

डायरेक्ट मैपिंग कैशे (Direct Mapping Cache) में मेन मेमोरी को n बिट्स के एड्रैस (Address) द्वारा और कैशे मेमोरी को k बिट्स के एड्रैस (Address) द्वारा प्रयोग किया जाता है। अब हम यह देखेंगे कि कैशे में वड्र्स को किस

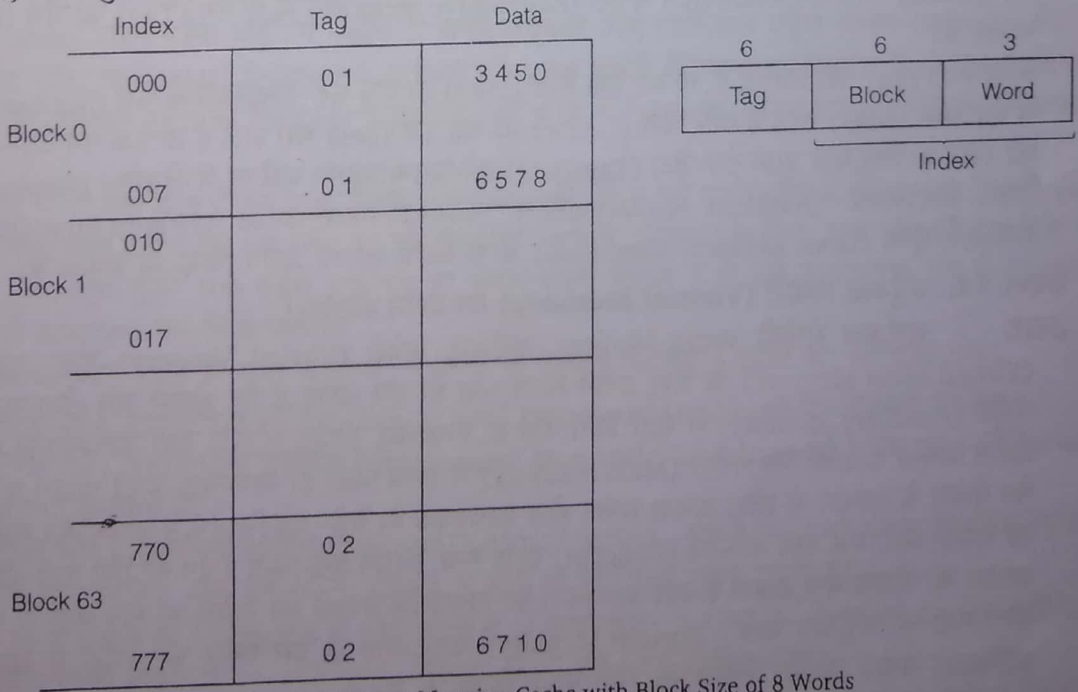
Scanned with CamScanner

प्रकार स्टोर (Store) किया जायेगा। कैशे में स्टोर किये गये वर्ड्स (Words) और मेन मेमोरी (Main Memory) में स्टोर (Store) किये गये वर्ड्स की आन्तरिक संरचना (Internal Organisation) को पिछले चित्र में दर्शाया गया है। हम देख सकते हैं कि कैशे (Cache) में इन्डैक्स (Index) भाग को एड्रैसिंग (Addressing) के लिए और टैग (Tag) भाग आन्तरिक तालिका में प्रयुक्त किया गया है। जब सीपीयू (CPU) से किसी वर्ड (Word) की रिक्वेस्ट (Request) आती है, तो पहले इन्डैक्स (Index) भाग द्वारा, उसे कैशे (Cache) में खोजा जाता है, फिर टैग (Tag) भाग में तुलना की जाती है और यदि वह टैग (Tag) कैशे (Cache) में प्राप्त हो जाता है, तो यह एक हिट (Hit) है और यदि नहीं मिलता है तो ये एक मिस (Miss) है और डेटा (Data) को मेन मेमोरी (Main Memory) से लाया जाता है और उस वर्ड (Word) को कैशे (Cache) में नये टैग (Tag) के साथ स्टोर (Store) कर दिया जाता है।

कैशे मेमोरी की कमी (Disadvantage) यह है, कि इसमें हिट रेशियो (Hit Ratio) बहुत तेजी से कम हो सकता है, क्योंकि एक ही इन्डैक्स (Index) के लिए अलग-अलग टैग्स (Tags) को एक साथ मेमोरी (Memory) में नहीं रखा जा सकता है।

जीरो एड्रैस (Zero Address) पर स्टोर किया गया वर्ड (Word) इस समय कैशे (Cache) में भी सुरक्षित है मान लेते हैं कि अब सीपीयू (CPU) 02000 पर सुरक्षित डेटा (Data) का प्रयोग करना चाहता है, तो दोनों अर्थात् जीरो एड्रैस (Zero Address) और 02000, के इन्डैक्स (Index) समान हैं। इस परिस्थिति में पहले tag 00 से तुलना की जाती है, जो कि समान नहीं है, फिर डेटा (Data) को मेन मेमोरी (Main Memory) से लाया जाता है और एड्रैस (Address) की तुलना की जाती है, जो कि वांछित एड्रैस (Address) से मिलता है, अतः इसके एड्रैस (Address) को रीड (Read) कर लिया जाता है और 6710 को सीपीयू (CPU) में भेज दिया जाता है। यहाँ हम देख सकते हैं कि एक ही इन्डैक्स (Index) के लिए अनेक टैग्स (Tags) उपलब्ध हैं, जिससे हिट रेशियो (Hit Ratio) काफी गिर जाता है।

इस समस्या के समाधान करने के लिए इन्डैक्स (Index) को दो फील्ड्स में विभाजित किया जाता है—ब्लॉक फील्ड (Block Field) और वर्ड फील्ड (Word Field)। 512-वर्ड की कैशे (512-Word Cache) में 8 वर्ड प्रति ब्लॉक (Block) के अनुसार 64 ब्लाक्स (Blocks) बनाये जा सकते हैं। निम्नांकित चित्र 2.10 देखें—



चित्र **2.10** Direct Mapping Cache with Block Size of 8 Words

Scanned with CamScanner

ब्लॉक (Block) को 6-Bit और वर्ड (Word) को 3-Bit में दिये गये हैं और टैग (Tag) को कैशे (Cache) के अन्दर प्रयोग किया गया है। अब दिये गये ब्लॉक नम्बर के आधार पर पहले ब्लॉक पर पहुँचेंगे, फिर वर्ड (Word) की तुलना करेंगे और उसके बाद टैग (Tag) की तुलना की जायेगी। इसके द्वारा हिट रेशियो (Hit Ratio) को बढ़ाया जा सकता है।

प्रश्न 10. सेट-एसोसिएटिव मैपिंग (Set-associative Mapping) को संक्षेप में समझाइए।

उत्तर **सेट-एसोसिएटिव मैपिंग** Set-associative Mapping डायरेक्ट मैपिंग (Direct Mapping) की कमी (Disadvantage) यह है कि इसमें एक ही इन्डैक्स (Index) के लिए दो विभिन्न टैग्स (Tags) का प्रयोग नहीं किया जा सकता है। सेट-एसोसिएटिव मैपिंग (Set-associative Mapping), डायरेक्ट मैपिंग (Direct Mapping) को सुधरा हुआ रूप है। इसमें कैशे (Cache) का प्रत्येक वर्ड (Word) समान इन्डैक्स (Index) के लिए दो अथवा दो से अधिक वर्ड्स (Words) को स्टोर (Store) कर सकता है। इसमें एक वर्ड (Word) में टैग (Tag) और उसका डेटा के सेट (Set) को स्टोर (Store) किया जा सकता है। उदाहरण के लिए, निम्नांकत चित्र 2.11 देखें।

Index	Tag	Data	Tag	Data
000	01	3450	02	5670
777	02	6710	00	2340

चित्र 2.11 Two-Way Set Associative Mapping Cache

उपरोक्त चित्र में हमने टू-वे सेट-एसोसिएटिव मैपिंग कैशे (Two Way Set-associative Mapping Cache) को प्रदर्शित किया है। इसमें जब सीपीयू (CPU) मेमोरी (Memory) रिक्वेस्ट (Request) करता है तो, एड्रैस (Address) के इन्डैक्स (Index) भाग से कैशे (Cache) का प्रयोग किया जाता है। तत्पश्चात् मैच (Match) देखने के लिए टैग फिल्ड (Tag Field) की दोनों टैग्स (Tags) से तुलना की जाती है और यदि मैच (Match) मिल जाता है तो, यह हिट (Hit) है। इस मेमोरी (Memory) को सेट-एसोसिएटिव (Set-Associative) इसलिए कहा जाता है, क्योंकि इसमें एक वर्ड (Word) के समान एसोसिएटिव मेमोरी (Associative Memory) में भी टैग (Tag) को मैच (Match) किया जा सकता है।

अगर टैग (Tag) नहीं मिलता है, तो यह एक मिस (Miss) है। जब सेट-एसोसिएटिव कैशे (Set-associative Cache) में एक मिस (Miss) होता है और कैशे (Cache) का वह सेट (Set) भरा होता है तो एक नये टैग-डेटा (Tag-Data) को रखने के लिए एक पुराने टैग-डेटा (Tag-Data) को हटाना पड़ेगा। यहाँ पर रिप्लेसमेन्ट (Replacement) के लिए रेन्डम रिप्लेसमेन्ट (Random Replacement), फर्स्ट-इन-फर्स्ट-आउट (First-In-First-Out-FIFO), लीस्ट रीसेन्टली यूज्ड (Least Recently Used LRU) में से किसी को भी प्रयोग किया जा सकता है।

प्रश्न 11. वर्चुअल मेमोरी (Virtual Memory) का वर्णन कीजिए।

उत्तर **वर्चुअल मेमोरी** Virtual Memory वर्चुअल मेमोरी (Virtual Memory) शब्द वर्चुअल मेन मेमोरी (Virtual Main Memory) के लिए प्रयोग किया गया है। हम जानते हैं कि प्रोग्राम और डेटा सर्वप्रथम ऑग्जिलरी मेमोरी (Auxiliary Memory) में स्टोर किये जाते हैं, तत्पश्चात् सीपीयू (CPU) द्वारा आवश्यकता होने पर आवश्यक प्रोग्राम अथवा डेटा को मेन मेमोरी (Main Memory) में लाया जाता है। सामान्यत: हमारे प्रोग्राम का आकार (Size), मेन मेमोरी के आकार से छोटा अथवा समान होना आवश्यक है। बड़े-बड़े सिस्टम में प्रोग्राम मेन मेमोरी पर निर्भर नहीं रह सकते; अत: एक ऐसा कॉन्सेप्ट (Concept) लाया गया, जिसमें प्रयोगकर्ता प्रयोग की गयी बड़ी ऑग्जिलरी मेमोरी के आकार का प्रोग्राम बना सकता है और प्रयोगकर्ता को लगेगा कि उसकी मेन मेमोरी का माप बड़ा हो गया है। इस प्रकार की मेमोरी को वर्चुअल मेमोरी (Virtual Memory) कहा जाता है। इस प्रकार की मेमोरी में आवश्यकता होने पर ऑग्जिलरी मेमोरी को मेन मेमोरी की भाँति प्रयोग किया जा सकता है।

इस प्रकार की मेमोरी में जब सीपीयू (CPU) द्वारा किसी एड्रैस (Address) का रेफरेन्स (Reference) दिया जाता है, तो वर्चुअल एड्रैस (Virtual Address) अर्थात् वर्चुअल मेमोरी (Virtual Memory) का एड्रैस (Address), की एड्रैस

Scanned with CamScanner

मैपिंग (Address Mapping) द्वारा मेन मेमोरी (Main Memory) का फिजिकल एड्रैस (Physical Address) प्रा किया जाता है। इस प्रकार प्रोग्रामर के सामने यह भ्रम उत्पन्न किया जाता है कि उसके पास बड़ी मेन मेमोरी (Ma Memory) है, जबकि उसके पास अपेक्षाकृत बहुत छोटी मेन मेमोरी होती है।

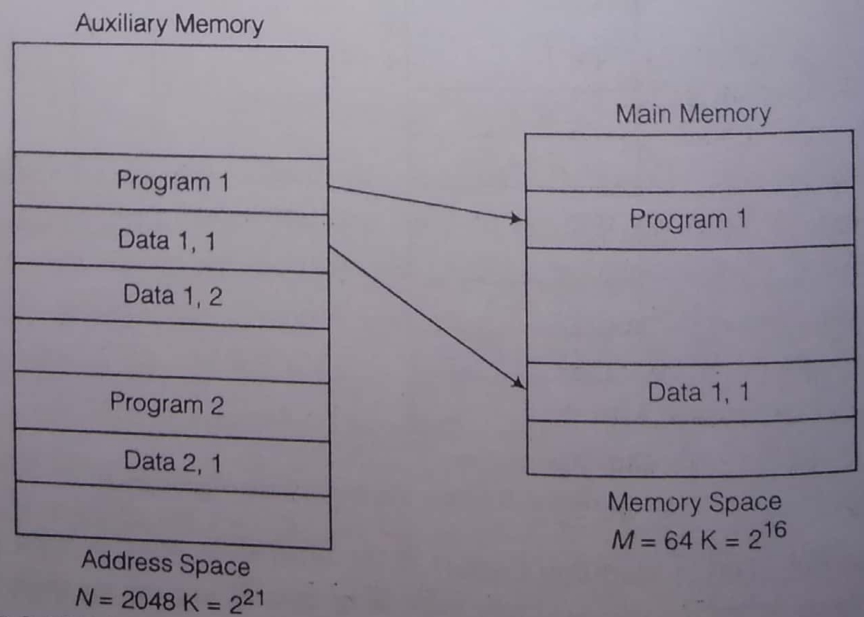
वर्चुअल मेमोरी सिस्टम (Virtual Memory System) प्रोग्राम द्वारा उत्पन्न किये गये एड्रैसेज (Program Generat Address) को करेक्ट मेन मेमोरी लोकेशन्स में ट्रान्सलेट (Translate) करने के लिए एक मेकेनिज्म उपलब्ध करा जाती है। इस ट्रान्सलेशन (Translation) अथवा मैपिंग (Mapping) को मैपिंग टेबल (Mapping Table) के माध्य से हार्डवेयर (Hardware) द्वारा स्वत: ही हैण्डल (Handle) किया जाता है।

वर्चुअल मेमोरी (Virtual Memory) के सन्दर्भ में निम्नलिखित शब्दों का बहुतायत में प्रयोग किया जाता है—

1. **वर्चुअल एड्रैस** Virtual Address प्रोग्रामर के द्वारा प्रयोग किये गये एड्रैस (Address) को वर्चुअल एड्रैस (Vir Address) कहा जाता है।
2. **एड्रैस स्पेस** Address Space वर्चुअल एड्रैसेज (Virtual Addresses) के समूह अर्थात् सेट (Set) को ए स्पेस (Address Space) कहा जाता है।
3. **फिजिकल एड्रैस अथवा लोकेशन** Physical Address Or Location मेन मेमोरी (Main Memory) के ए (Address) को फिजिकल एड्रैस अथवा लोकेशन (Physical Address Or Location) कहा जाता है।
4. **मेमोरी स्पेस** Memory Space फिजिकल एड्रैसेज (Physical Addresses) के समूह अर्थात् सेट (Set) मेमोरी स्पेस (Memory Space) कहा जाता है।

यदि सिस्टम में वर्चुअल मेमोरी (Virtual Memory) का प्रयोग नहीं किया गया है, तो एड्रैस स्पेस (Address Spac की माप मेमोरी स्पेस (Memory Space) की माप के समान ही है। अधिकतर सिस्टम में एड्रैस स्पेस (Addre Space) और मेमोरी स्पेस (Memory Space) समान ही होते हैं। यदि सिस्टम में वर्चुअल मेमोरी (Virtual Memor का प्रयोग किया गया है तो एड्रैस स्पेस (Address Space) की माप मेमोरी स्पेस (Memory Space) की माप अपेक्षाकृत अधिक हो सकती है।

उदाहरण के लिए, मान लेते हैं कि एक कम्प्यूटर की मेन मेमोरी (Main Memory) की क्षमता 32K (K = 102 वर्ड्स (Words) की है। चूँकि $2^{16} = 64K$ है, इसलिए इस मेमोरी के एड्रैस (Address) के लिए 15 बिट्स आवश्यकता होगी। मान लेते हैं कि इस कम्प्यूटर की ऑग्जिलरी मेमोरी (Auxiliary Memory) की क्षमता 1024K चूँकि $2^{21} = 2048K$ है, इसलिए इस मेमोरी के एड्रैस (Address) के लिए 20 बिट्स की आवश्यकता होगी। चूँ $32 * 64K = 2048K$, अत: हम एक ऑग्जिलरी मेमोरी (Auxiliary Memory) की सहायता 32 मेन मेमोरी (M Memory) के समान क्षमता की मेमोरी का प्रयोग कर सकते हैं।



चित्र 2.12 Relation Between Address and Memory Space in a Virtual Memory System

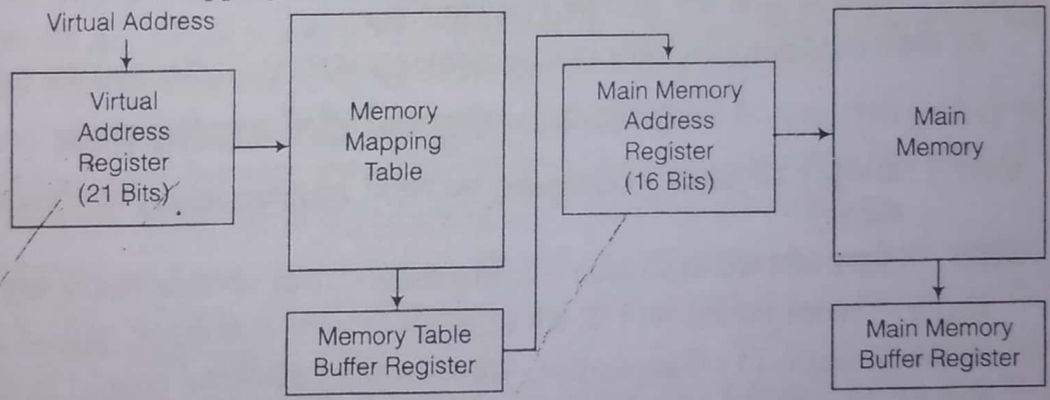
Scanned with CamScanner

यदि एड्रैस स्पेस (Address Space) को N से व मेमोरी स्पेस (Memory Space) को M से प्रदर्शित करें, तो इस उदाहरण के लिए $N = 2048K$ और $M = 64K$।

एक मल्टीप्रोग्राम कम्प्यूटर सिस्टम में, सीपीयू (CPU) की मांग (Demand) पर प्रोग्राम और डेटा को ऑग्जिलरी मेमोरी से मेन मेमोरी (Main Memory) में और मेन मेमोरी से ऑग्जिलरी मेमोरी में भेजा जाता है। उदाहरण के लिए, चित्र 2.12 देखें।

वर्चुअल मेमोरी (Virtual Memory) में यदि Program1 को एग्जिक्यूट (Execute) किया जा रहा हो, तो program 1 और इससे जुड़े हुए आवश्यक भागों को ऑग्जिलरी मेमोरी (Auxiliary Memory) से मेन मेमोरी (Main Memory) में भेजा जाता है।

वर्चुअल मेमोरी सिस्टम (Virtual Memory System) में, प्रोग्रामर समस्त एड्रैस स्पेसेज (Address Space) का उपयोग कर सकते हैं। हमारे उदाहरण में प्रोग्रामर एड्रैस स्पेस (Address Space) का प्रयोग कर रहे हैं तो एड्रैस (Address) 21 बिट्स का होगा और फिजिगल एड्रैस (Physical Address) के लिए केवल 16 बिट्स की आवश्यकता होती है। परन्तु जब सीपीयू (CPU) किसी निर्देश और डेटा का रेफरेन्स (Reference) देगा, तो उसका एड्रैस (Address) 21 बिट्स का होगा, परन्तु इस एड्रैस (Address) पर दी गयी सूचना को पहले मेन मेमोरी (Main Memory) में भेजा जायेगा, जहाँ से सीपीयू (CPU) उस सूचना का उपयोग करेगा। इस कार्य को करने के लिए एक तालिका अर्थात् टेबल (Table) की आवश्यकता होती है, इस तालिका के द्वारा 21 बिट्स के वर्चुअल एड्रैस (Virtual Address) को 16 बिट के फिजिकल एड्रैस (Physical Address) में परिवर्तित किया जा सकता है। इस तालिका के द्वारा सीपीयू (CPU) के द्वारा रेफरेन्स (Reference) किये गये एड्रैस (Address) को तुरन्त परिवर्तित किया जा सकता है। इस मैपिंग तालिका (Mapping Table) के लिए चित्र 2.13 को देखें—



चित्र **2.13** Memory Table for Mapping a Virtual Address

मैपिंग टेबल (Mapping Table) को पृथक् मेमोरी (Separate Memory) अथवा मेन मेमोरी (Main Memory) में स्टोर (Store) किया जा सकता है। यदि हम पृथक् मेमोरी (Separate Memory) का प्रयोग करना चाहते हैं तो इसके लिए एक अतिरिक्त मेमोरी यूनिट (Additional Memory Unit) की आवश्यकता होगी और साथ ही अधिक मेमोरी एक्सेस टाइम (Memory Acess Time) भी लगेगा।। यदि मेन मेमोरी (Main Memory) का प्रयोग करेंगे, तो मेन मेमोरी (Main Memory) पर दो बार कार्य करना पड़ेगा, अतः प्रोग्राम आधी स्पीड (गति) पर कार्य करेगा। तीसरा तरीका है एसोसिएटिव मेमोरी (Associative Memory) का प्रयोग।

प्रश्न 12. रीड ऑपरेशन (Read Operation) तथा राइट ऑपरेशन (Write Operation) पर टिप्पणी लिखिए।

उत्तर **रीड ऑपरेशन** Read Operation वह ऑपरेशन जिसके द्वारा किसी मेमोरी लोकेशन पर स्टोर किये गए बाइनरी वर्ड को Sense करके किसी अन्य डिवाइस को ट्रांसफर किया जाता है। मेमोरी रीड ऑपरेशन कहलाता है। रीड ऑपरेशन को Fetch ऑपरेशन भी कहा जाता है।

राइट ऑपरेशन Write Operation मेमोरी की किसी लोकेशन पर नये बाइनरी वर्ड को प्लेस करना अर्थात् किसी मेमोरी लोकेशन पर बाइनरी वर्ड को स्टोर करना राइट ऑपरेशन कहलाता है। जब भी किसी मेमोरी लोकेशन पर नया बाइनरी वर्ड राइट किया जाता है तो वह उस लोकेशन पर पहले से स्टोर बाइनरी वर्ड को रिपलेस कर देता है।

Scanned with CamScanner

प्रश्न 13. मेमोरी मैनेजमेंट क्या है? प्रोसेस एड्रैस स्पेस को उदाहरण सहित समझाइए।

उत्तर **मेमोरी मैनेजमेंट** Memory Management मेमोरी मैनेजमेंट एक ऑपरेटिंग सिस्टम (Operating System) का फंक्शन है जो प्राइमरी मेमोरी को हैंडल करता है और मैनेज करता है। इसके साथ ही ये प्रोसेस को Execution के दौरान मेमोरी और डिस्क के बीच घुमाने का काम भी करता है। मेमोरी मैंनेजमेंट सारे के सारे मेमोरी लोकेशन का ट्रैक रखता है; भले ही उन्हें किसी प्रोसेस को Allocate किया गया हो या फिर वो खाली बैठे हों। ये इस बात की जाँच भी करता है कि किस प्रोसेस को कितनी मेमोरी देने की जरूरत है। कौन-से प्रोसेस को किस समय पर मेमोरी दी जाएगी। ये तय करना भी मेमोरी मैनेजमेंट का ही कार्य होता है।

जब भी कोई मेमोरी फ्री होती है अर्थात् किसी प्रोसेस को एक्सीक्यूट करने के बाद खाली होता है तो मेमोरी मैनेजमेंट उसी हिसाब से अपने रिकॉर्ड को अपडेट भी करता रहता है।

यहाँ हम मेमोरी मैनेजमेंट से जुड़ी जरूरी बातों के साथ-साथ उसकी प्रक्रिया को भी अच्छी तरह से समझेंगे।

प्रोसेस एड्रैस स्पेस Process Address Space प्रोसेस एड्रैस स्पेस कुछ लॉजिकल एड्रैस का एक सेट है और ये वो एड्रैस होते हैं जिसका रिफरेन्स प्रोसेस के कोड में होता है।

उदाहरणार्थ जब 32 बिट एड्रैसिंग का प्रयोग किया जा रहा हो तो एड्रैस का रेंज 0 से 0 × 7fffffff तक हो सकता है अर्थात् दो गीगाबाइट के थ्योरिटिकल आकार के लिए 2^{31} सम्भव संख्याएँ हो सकती हैं।

ऑपरेटिंक सिस्टम प्रोग्राम में मेमोरी एलोकेशन के समय लॉजिकल एड्रैस को फिजिकल एड्रैस से मैप करने का भी ध्यान रखता है।

किसी भी प्रोग्राम में मेमोरी के एलोकेशन होने से पहले और बाद में कुल तीन तरह के एड्रैस का प्रयोग होता है—

1. **सिंबॉलिक एड्रैस** इस एड्रैस को सोर्स कोड में प्रयोग किया जाता है। वेरिएबल के नाम, Constraint और इंस्ट्रक्शन लेबल्स—ये सिंबॉलिक एड्रैस के मूल एड्रैस स्पेस होते हैं।
2. **रिलेटिव एड्रैस** कंपाइलेशन वाली प्रक्रिया के समय पर एक कम्पाइलर सिंबॉलिक एड्रैस को रिलेटिव एड्रैस में बदलने का काम करता है।
3. **फिजिकल एड्रैस** लोडर इन एड्रैस को उस समय Generate करता है जब प्रोग्राम मेमोरी के अंदर लोड होता है।

कम्पाइल टाइम और लोड टाइम एड्रैस बाइंडिंग स्कीम के समय पर वर्चुअल और फिजिकल एड्रैस दोनों समान ही होते हैं। वहीं Execution टाइम एड्रैस बाइंडिंग स्कीम के समय वर्चुअल और फिजिकल एड्रैस अलग-अलग होते हैं। मेमोरी मैनेजमेंट यूनिट यानी MMU फिजिकल से वर्चुअल एड्रैस तक मेमोरी मैपिंग का कार्य करता है। ये एक हार्डवेयर डिवाइस होता है। वर्चुअल एड्रैस को फिजिकल एड्रैस में कन्वर्ट करने के लिए MMU निम्नलिखित मैकेनिज्म का प्रयोग करता है—

- बेस रजिस्टर में जो मान होता है उसे यूजर प्रोसेस द्वारा Generate किये गये सभी एड्रैस से जोड़ा जाता है। जब इसे मेमोरी को भेजा जाता है जब इसे एक ऑफसेट की तरह देखा जाता है। उदाहरण के लिए, अगर बेस रजिस्टर में मान है 10000, तो अगर यूजर 100 एड्रैस लोकेशन का प्रयोग करना चाहता है तो उसे डायनामिक रूप से 10100 लोकेशन को Allocate कर दिया जायेगा।
- यूजर प्रोग्राम वर्चुअल एड्रैस से डील करता है, ये कभी भी असली फिजिकल एड्रैस को नहीं देख पाता।

प्रश्न 14. स्टैटिक और डायनामिक लोडिंग (Static and Dynamic Loading) पर टिप्पणी लिखिए।

अथवा **स्टैटिक और डायनामिक लिंकिंग (Static and Dynamic Linking) को समझाइए। स्वैपिंग से आप क्या समझते हैं?**

उत्तर **स्टैटिक और डायनामिक लोडिंग** Static and Dynamic Loading जब कम्प्यूटर का प्रोग्राम विकसित किया जा रहा होता है तभी स्टैटिक और डायनामिक लोडिंग के बीच से कोई एक चुनना पड़ता है।

अगर आपको अपने प्रोग्राम को स्टैटिक रूप से लोड करना है तो कंपाइलेशन के समय पूरे प्रोग्राम को Compile किया जाएगा और बिना किसी बाहरी प्रोग्राम और मोडुले निर्भरता के ही इसे लिंक किया जाएगा।

लिंकर ऑब्जेक्ट प्रोग्राम को किसी और जरूरी ऑब्जेक्ट Modules से Absolute प्रोग्राम के अंदर लिंक करता है जिसमें लॉजिकल एड्रैस भी शामिल होता है।

Scanned with CamScanner

अगर आप डायनामिक रूप से लोड होने वाला प्रोग्राम लिख रहे हैं तो आपका कम्पाइलर सभी वैसे Modules के लिए प्रोग्राम को Compile करेगा जिसे आप Dynamically जोड़ना चाहते हैं। केवल रिफरेन्स दिया जाएगा और बाकी का काम Execution के समय पर पूरा होगा। जब लोडिंग हो रहा हो तब स्टैटिक लोडिंग के समय, Absolute प्रोग्राम और डाटा को भी मेमोरी में लोड किया जाता है ताकि Execution शुरू हो जाये। अगर आप डायनामिक लोडिंग का प्रयोग कर रहे हैं तो लाइब्रेरी के डायनामिक रूटीन को एक डिस्क में ऐसे स्टोर किया जाएगा कि जब जरूरत पड़े तो उसे कहीं और फिर से लोकेट किया जा सके। उन्हें मेमोरी में तभी लोड किया जाता है जब प्रोग्राम को उनकी जरूरत हो।

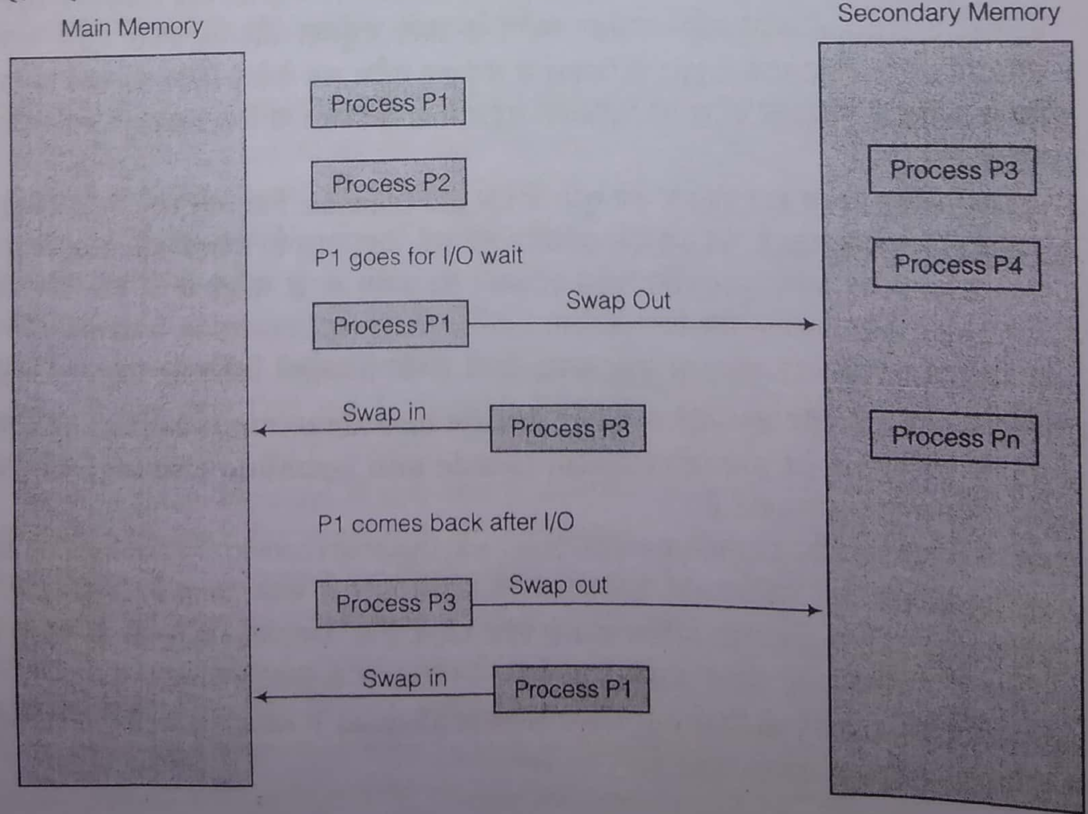
स्टैटिक और डायनामिक लिंकिंग Static and Dynamic Linking जैसा कि आपने ऊपर देखा, जब स्टैटिक लिंकिंग का प्रयोग किया जाता है तब लिंकर प्रोग्राम को जिनकी जरूरत हो उन सारे Modules को आपस में कंबाइन करके एक ऐसा प्रोग्राम बना देता है जिसे एक्सीक्यूट किया जा सके और इसे करने के पीछे का कारण होता है रनटाइम निर्भरता को हटाना।

वहीं दूसरी तरफ जब डायनामिक लिंकिंग का प्रयोग किया जाता है तब प्रोग्राम के साथ असली Module या लाइब्रेरी को जोड़ने की कोई जरूरत नहीं होती।

इसके बजाय कंपाइलेशन और लिंकिंग के समय डायनामिक Module का एक रेफरेन्स दे दिया जाता है। डायनामिक लाइब्रेरी के प्रमुख उदाहरण हैं—विंडोज में डायनामिक लिंक लाइब्रेरी (DLL) और यूनिक्स में शेयर्ड ऑब्जेक्ट्स।

स्वैपिंग Swapping in Memory Management in Hindi स्वैपिंग एक ऐसा मैकेनिज्म है जिसमें किसी भी प्रोसेस को अस्थाई रूप मेन मेमोरी से सेकेंडरी स्टोरेज यानी कि डिस्क में स्वैप किया जा सकता है और उस मेमोरी को अन्य प्रोसेस के लिए उपलब्ध कराया जा सकता है। कुछ समय के बाद, सिस्टम फिर से प्रोसेस को डिस्क में से अपने मेन मेमोरी में वापस स्वैप कर लेता है।

वैसे तो स्वैपिंग की प्रक्रिया से सिस्टम के परफॉरमेन्स पर असर पड़ता है लेकिन ये बड़े और एक से ज्यादा प्रोसेस को रन करने में मदद करता है। यही कारण है कि स्वैपिंग को मेमोरी Compaction की तक्तिक भी कहा गया है। स्वैपिंग प्रक्रिया द्वारा लिए गए कुल समय में इसके द्वारा मेन मेमोरी से डिस्क में प्रोसेस को मूव करने में जो समय लगता है वो भी शामिल होता है।



Scanned with CamScanner

3 गणितीय संक्रियाएँ
Arithmetic Operations

खण्ड 'अ' : अतिलघु उत्तरीय प्रश्न

प्रश्न 1. डिजिटल कम्प्यूटर्स में किन निर्देशों पर कार्य किया जाता है?

उत्तर डिजिटल कम्प्यूटर्स में अर्थमैटिक निर्देशों पर कार्य किया जाता है।

प्रश्न 2. गणित में कितने मुख्य ऑपरेशन होते हैं?

उत्तर गणित में चार मुख्य ऑपरेशन होते हैं।

प्रश्न 3. कम्प्यूटर में किसी समस्या के हल को प्राप्त करने के लिए उपयोग में आने वाले सीमित स्टेप्स (को क्या कहते हैं?

उत्तर एल्गोरिथ्म।

प्रश्न 4. गणित के मुख्य चार ऑपरेशनों के नाम लिखिए।

उत्तर गणित के मुख्य चार ऑपरेशन निम्नलिखित हैं—

(i) जोड़ना (Addition) (ii) घटाना (Subtraction) (iii) गुणन (Multiplication) तथा (iv) विभाजन (Divisi

खण्ड 'ब' : लघु एवं दीर्घ उत्तरीय प्रश्न

प्रश्न 1. कम्प्यूटर की जोड़ने (Addition) एवं घटाने (Subtraction) की क्रियाओं का विस्तारपूर्व कीजिए।

उत्तर **जोड़ना और घटाना** Addition and Subtraction किसी ऋणात्मक फिक्स्ड प्वॉइन्ट बायनरी (Fixe Binary) अंक को निरूपित करने के तीन तरीके होते हैं—साइन्ड-मैग्नीट्यूड (Signed-Magnitude), सा काम्प्लीमेन्ट (Signed 1's Complement) अथवा साइन्ड 2's काम्प्लीमेन्ट (Signed 2's Complement)। इ कम्प्यूटर जब इन्टीजर (Integer) के साथ अर्थमेटिक (Arithmetic) कार्य करते हैं, तो वे साइन्ड 2's का (Signed 2's Complement) का प्रयोग करते हैं। अधिकतर कम्प्यूटर फ्लोटिंग प्वॉइन्ट ऑपरेशन (Floatin Operation) में मेन्टिसा (Mantissa) अर्थात् रीयल नम्बर के फ्रेक्शनल भाग (The Fractional Part of Number), को निरूपित करने के लिए साइन्ड-मैग्नीट्यूड (Signed-Magnitude) का प्रयोग करते हैं।

साइन्ड मैग्नीट्यूड डेटा के साथ जोड़ना और घटाना Addition and Subtraction with Signed Mag Data साइन्ड मैग्नीट्यूड (Signed Magnitude) का तात्पर्य है अंकों के आगे धन (Plus) तथा ऋण (Min चिह्न प्रदर्शित करना। अतः हम यह कह सकते हैं कि यह हमारे प्रतिदिन होने वाली गणित की गणनाओं के ही स चिह्नों के साथ बायनरी (Binary) अंकों को कागज पर जोड़ना और घटाना सरल है।

अब हम दो नम्बर्स (Numbers) के परिमाण अर्थात् मैग्नीट्यूड (Magnitude) को A और B से प्रदर्शित करेंगे। चिह्नित नम्बर (Number) को जोड़ा जाता है, तो चिह्न और ऑपरेशन (Operation) के आधार पर कुल आठ स्थितियाँ बनती हैं, जिन्हें अग्रांकित तालिका में प्रदर्शित किया गया है—

Operation	Add Magnitudes	Subtract Magnitudes		
		When $A > B$	When $A < B$	When $A = B$
$(+A) + (+B)$	$+(A+B)$			
$(+A) + (-B)$		$+(A-B)$	$-(B-A)$	$+(A-B)$
$(-A) + (+B)$		$-(A-B)$	$+(B-A)$	$+(A-B)$
$(-A) + (-B)$	$-(A+B)$			
$(+A) - (+B)$		$+(A-B)$	$-(B-A)$	$+(A-B)$
$(+A) - (-B)$	$+(A+B)$			
$(+A) - (-B)$	$-(A+B)$			
$(+A) - (-B)$		$-(A-B)$	$+(B-A)$	$+(A-B)$

यहाँ पहले कॉलम (Column) में हमने आठों स्थितियों को प्रदर्शित किया है और शेष कॉलम्स में वास्तव में होने वाले ऑपरेशन (Operation) को प्रदर्शित किया गया है। इस तालिका के आधार पर जोड़ने और घटाने के लिए एल्गोरिथ्म (Algorithm) को व्युत्पन्न (Derived) किया जा सकता है। निम्नलिखित एल्गोरिथ्म (Algorithm) में कोष्ठक में दिए संकेत घटा के लिए हैं।

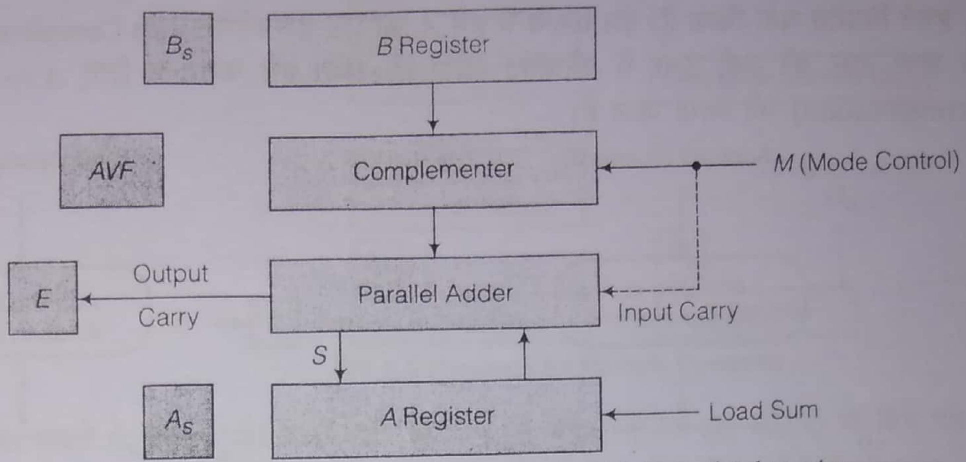
जोड़ना (घटाना) एल्गोरिथ्म [(Addition (Subtraction) Algorithm] जब A और B के चिह्न समान हैं, तो दो नम्बर्स को जोड़ करें और उत्तर पर A का चिह्न लगायें। जब A और B के चिह्न विभिन्न हैं, तो उनके मैग्नीट्यूड (Magnitude) की तुलना करें और बड़े नम्बर में से छोटे नम्बर को घटायें। यदि $A > B$ है तो उत्तर के सामने A का चिह्न लगायें अथवा यदि $A < B$ है, तो उत्तर के सामने A के कॉम्पलीमेन्ट (Complement) का चिह्न लगायें। यदि दोनों के (Magnitude) समान हैं, तो B में से A को घटायें और उस पर धनात्मक चिह्न लगायें।

जोड़ने (Addition) और घटाने (Subtraction), दोनों की एल्गोरिथ्म समान होती है। अन्तर सिर्फ तुलनाओं का होता है, जहाँ जोड़ने (Addition) में चिह्न समान होने चाहिए, वहाँ घटान (Subtraction) में भिन्न चिह्न और जहाँ जोड़ने (Addition) में भिन्न होने चाहिये, वहाँ घटाने (Subtraction) में समान चिह्न होने चाहियें।

हार्डवेयर इम्प्लीमेन्टेशन Hardware Implementation इन दोनों ऑपरेशन्स (Operations) को हार्डवेयर द्वारा कार्यान्वित करने के लिए, दोनों अंकों और उनके चिह्न को सुरक्षित करना अनिवार्य होता है। माना कि A और B दो रजिस्टर्स (Registers) हैं, जिनमें अंकों को सुरक्षित किया जायेगा और A_s व B_s दो फ्लिप-फ्लॉप (Flip-Flop) हैं, जिनमें अंकों के चिह्नों को सुरक्षित किया जायेगा। ऑपरेशन के उत्तर को एक तीसरे रजिस्टर में सुरक्षित किया जा सकता है, परन्तु यहाँ हम A और A_s का प्रयोग उत्तर के सुरक्षित करने के लिए भी करेंगे।

अब उपरोक्त एल्गोरिथ्म (Algorithm) के अनुसार हम ऑपरेशन (Operation) को हार्डवेयर द्वारा कार्यान्वित करेंगे। इसके लिए पहले $A + B$ माइक्रोऑपरेशन (Microoperation) के लिए एक समान्तर-एडर (Parallel Adder) की आवश्यकता होगी। दूसरे, यह जाँचने के लिए कि $A > B$ अथवा $A < B$ अथवा $A = B$, एक तुलनाकारी सर्किट (Comparator Circuit) की आवश्यकता होगी। तीसरे, $A - B$ अथवा $B - A$ माइक्रोऑपरेशन (Microoperation) के लिए एक समानान्तर सबट्रैक्टर (Parallel-Subtractor) की आवश्यकता होगी। चिह्नों के मध्य के सम्बन्ध को, एक एक्सक्लूसिव (Exclusive) OR गेट के साथ A_s और B_s को इनपुट की भाँति प्रवाहित करके प्राप्त कर सकते हैं। इस प्रक्रिया में एक मैग्नीट्यूड कम्परेटर (Magnitude Comparater), एक एडर (Adder) और दो सबट्रैक्टर (Subtractor) की आवश्यकता होती है। एक भिन्न प्रक्रिया द्वारा कम युक्तियों द्वारा भी इस कार्य को सम्पन्न किया जा सकता है। पहले हम जानते हैं कि घटाने (Subtraction) के लिए जोड़ (Add) और काम्प्लीमेन्ट (Complement) का प्रयोग किया जा सकता है। दूसरे, तुलना को घटाने के बाद अन्तिम कैरी (Carry) द्वारा जांच सकते हैं। इस प्रकार सिर्फ एक एडर (Adder) और काम्प्लीमेन्टर (Complementer) की सहायता से इस कार्य को किया जा सकता है।
अग्रांकित चित्र 3.1 में हार्डवेयर द्वारा जोड़ने (Addition) और घटाने (Subtraction) को प्रदर्शित किया गया है—

Scanned with CamScanner



चित्र 3.1 Hardware for Signed-Magnitude Addition and Subtraction

यहाँ पर दो रजिस्टर्स (Registers) A और B एवं दो फ्लिप-फ्लाप (Flip-Flop) A_s और B_s हैं। घटाने (Subtraction) के लिए B के 2's काम्प्लीमेन्ट (2's Complement) को A के साथ जोड़ा जाता है। आउटपुट हासिल अर्थात् कैरी (Carry) को फ्लिप-फ्लाप (Flip-Flop) E में भेजा जाता है, जहाँ इसे दोनों मैग्नीट्यूड्स (Magnitudes) की तुलना के लिए प्रयोग किया जाता है। AVF फ्लिप-फ्लॉप, एड ओवरफ्लो (Add Overflow) के लिए प्रयोग किया गया है, जिसमें A और B को जमा करने के पश्चात् प्राप्त होने वाली ओवरफ्लो बिट (Overflow Bit) को सुरक्षित किया जाता है। दोनों अंकों के योग को S द्वारा A में सुरक्षित किया जाता है।

A और B का जोड़ (Addition) करने के लिए समानान्तर-एडर (Parallel-Adder) का प्रयोग किया जाता है। काम्प्लीमेन्टर (Complementer), मोड नियन्त्रण (Mode Control) के अनुसार B अथवा B के काम्प्लीमेन्ट (Complement) को आउटपुट की भाँति प्रवाहित करता है। मोड नियन्त्रण (Mode Control) M इनपुट कैरी (Input Carry) को नियन्त्रित करता है। यदि $M = 0$ है, तो B को आउटपुट की भाँति एडर (Adder) में प्रवाहित होगा और इनपुट कैरी (input carry) भी शून्य (0) होगी और एडर (Adder) का आउटपुट योग $A + B$ के समान होगा। जब $M = 1$ है, तो B का 1's काम्प्लीमेन्ट (1's Complement) आउटपुट की भाँति एडर (Adder) में प्रवाहित होगा और इनपुट कैरी (Input Carry) 1 होगी और आउटपुट $S = A + B + 1$ होगा। जिसका तात्पर्य है कि A तथा B का 2's कॉम्प्लीमेन्ट (2's Complement), जो कि $A - B$ के समान उत्तर प्रदान करता है।

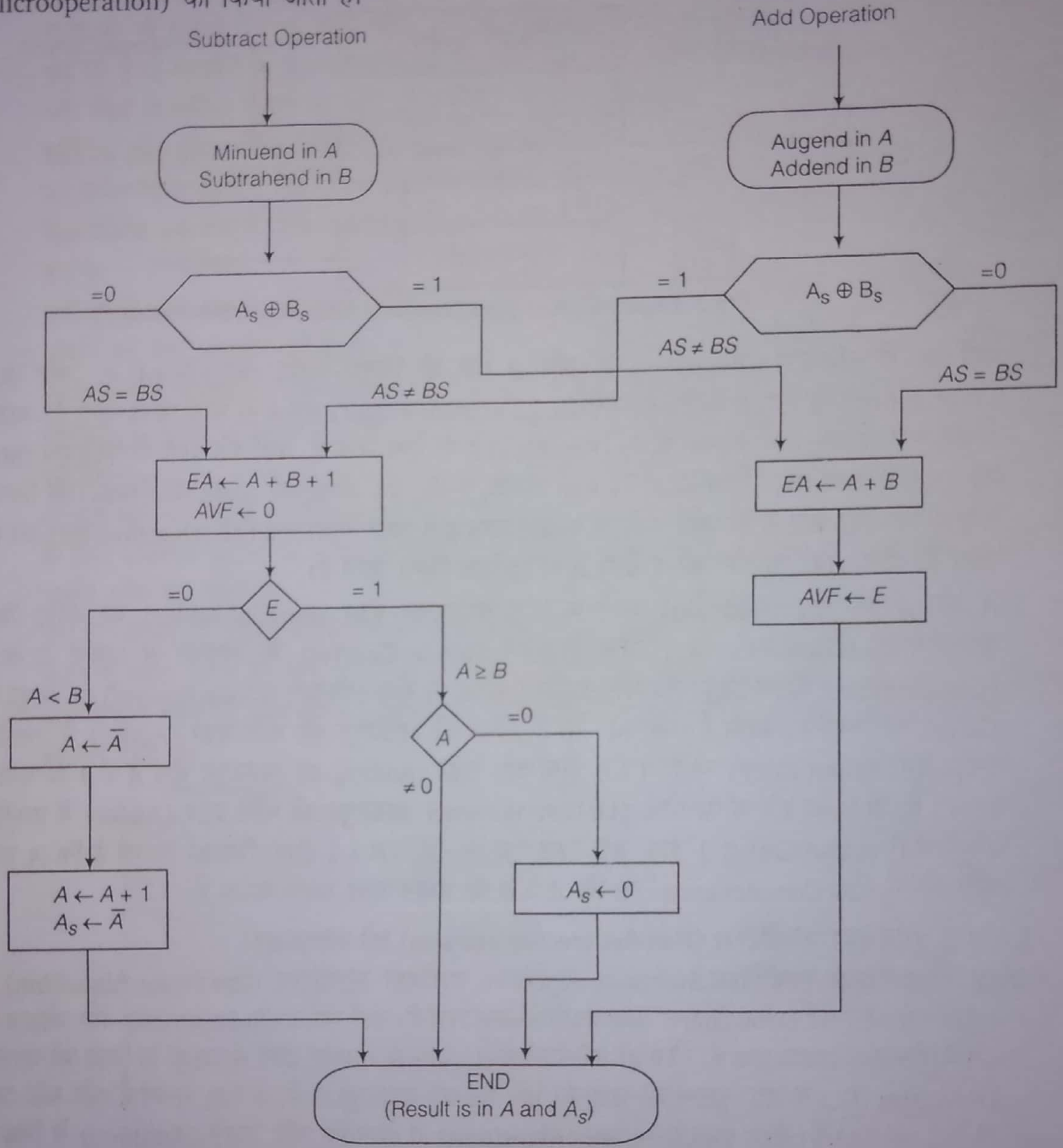
प्रश्न 2. हार्डवेयर एल्गोरिथ्म (Hardware Algorithm) को समझाइए।

उत्तर **हार्डवेयर एल्गोरिथ्म** Hardware Algorithm हार्डवेयर एल्गोरिथ्म (Hardware Algorithm) चित्र 3.2 में दर्शाये गये फ्लोचार्ट (Flowchart) द्वारा प्रदर्शित किया गया है। यहाँ घटाना (Subtraction) और जोड़ना (Addition) ऑपरेशन्स को दिखाया गया है। दोनों ही ऑपरेशन में A_s और B_s अर्थात् दोनों संख्याओं के चिह्नों की एक्सक्लूसिव OR (Exclusive OR) के द्वारा तुलना की जाती है। यदि गेट का आउटपुट 0 है, तो चिह्न समान है और यदि आउटपुट 1 है, तो चिह्न असमान है। जैसा हमने एल्गोरिथ्म (Algorithm) में देखा कि यदि जोड़ने (Addition) में चिह्न समान हैं, तो जोड़ (Add) होगा और यदि घटाना (Subtraction) में चिह्न भिन्न है तो भी जोड़ (Add) होगा। इस ऑपरेशन को माइक्रोऑपरेशन (Microoperation) $EA \leftarrow A + B$ द्वारा किया जाता है, जहाँ EA एक रजिस्टर है, जो E और A को एक साथ कर देता है। यदि $E = 1$ है, तो इसका तात्पर्य है कि जोड़ (Add) करने के पश्चात् हासिल अर्थात् (Carry) है।

E के मान को एड-ओवरफ्लो फ्लिप-फ्लॉप (Add Overflow Flip-Flop–AVF) में भेजा जाता है। दोनों मैग्नीट्यूड्स (Magnitudes) को तभी घटाया जायेगा, जब जोड़ (Add) ऑपरेशन (Operation) में चिह्न भिन्न व घटा (Subtraction) ऑपरेशन में चिह्न समान होंगे। $A - B$ करने के लिए A में B के 2's कॉम्प्लीमेन्ट (2's Complement) को जोड़ा जाता है। जब घटाया जाता है, तो कोई ओवरफ्लो (Overflow) नहीं होता है, इसलिए AVF को 0 कर दिया जाता है। $E = 1$ का तात्पर्य है $A \geq B$ और चूँकि हमने $A - B$ किया है, अतः A में स्थित अंक सही उत्तर हैं यदि A का मान 0 है तो A_s को धनात्मक बनाना होता है, जिससे ऋणात्मक 0 न मिले। $E = 0$ का तात्पर्य है $A \leq B$,

Scanned with CamScanner

अत: हमने विपरीत घटा किया है। इस स्थिति में हमें A का 2's काम्प्लीमेन्ट (2's Complement) प्राप्त करना होगा, ताकि प्राप्त उत्तर को सही उत्तर में परिवर्तित किया जा सके। इसे करने के लिए $A \leftarrow \bar{A} + 1$ माइक्रोऑपरेशन (Microoperation) को किया जाता है।



चित्र **3.2** Flow Chart for Subtraction and Addition

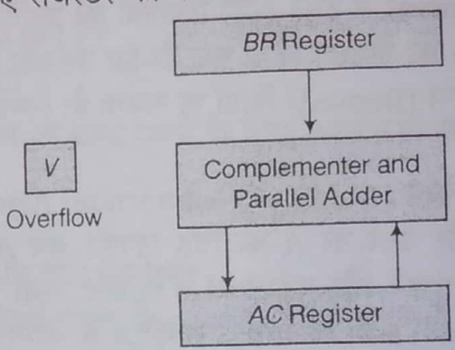
जैसा कि हम जानते हैं कि किसी भी अंक का 2's कॉम्प्लीमेन्ट (2's Complement) उस अंक के कॉम्प्लीमेन्ट (Complement) जमा 1 के समान होता है। *AVF* का मान ओवरफ्लो (Overflow) दर्शाता है।

प्रश्न 3. साइन्ड 2's (Signed 2's) कॉम्प्लीमेन्ट डेटा के साथ जोड़ने एवं घटाने को समझाइए।

उत्तर साइन्ड 2's **कॉम्प्लीमेन्ट डेटा के साथ जोड़ना और घटाना** Addition and Subtraction with Signed 2's Complement Data संख्याओं के साइन्ड 2's काम्प्लीमेन्ट (Signed 2's Complement Data) द्वारा निरूपण में बायनरी अंक की सबसे बायीं ओर वाली बिट के अंक के चिह्न को दर्शाती है। इसमें 0 का तात्पर्य है—धनात्मक और 1 का तात्पर्य है—ऋणात्मक। यदि चिह्न बिट (Bit) का मान 1 है, तो पूरी संख्या को उसके 2's काम्प्लीमेन्ट (2's Complement) के रूप में निरूपित किया जाता है। जैसे कि + 25 को 011001 द्वारा और – 25 को 100111 द्वारा

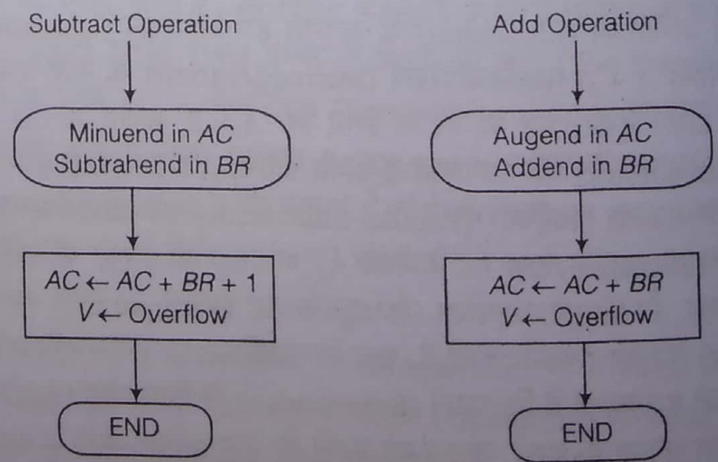
Scanned with CamScanner

निरूपित किया जाता है। ध्यान दें कि 100111 का 2's काम्प्लीमेन्ट (2's Complement) 011001 और 0
2's काम्प्लीमेन्ट (2's Complement) 100111 है।
साइन्ड 2's काम्प्लीमेन्ट (Signed 2's Complement) में जोड़ करते समय चिह्न बिट (Sign Bit) को अ
(Other Bits) की भाँति ही प्रयोग किया जाता है। चिह्न बिट (Sign Bit) के स्थान के हासिल अर्थात् कैरी (C
हटा देना चाहिए। घटाते समय उस संख्या का 2's काम्प्लीमेन्ट (2's Complement) ले लेना चाहिये जिसे
तत्पश्चात् इसे दूसरी संख्या के साथ जोड़ा जाता है।
जब n अंकों की दो संख्याओं को जोड़ा जाता है और योग में $n + 1$ अंक होते हैं, तो हम कहते हैं कि
(Overflow) हुआ है। ओवरफ्लो (Overflow) को जांचने के लिए जोड़ के अन्तिम हासिलों को जांचा ज
गेट (Gate) का आउटपुट 1 के समान होता है, तब दो हासिलों को एक्सक्लूसिव OR (Exclusive OR)
किया जाता है और ओवरफ्लो (Overflow) को जांच लिया जाता है।
हार्डवेयर द्वारा कार्यान्वित करने के लिए रजिस्टर की रूपरेखा निम्नांकित चित्र 3.3 में दर्शायी गयी है—



चित्र 3.3 Hardware for Signed-2's Complement Addition and Subtraction

उपरोक्त चित्र 3.3 पृष्ठ 45 पर दिए गये चित्र 3.1 के समान ही है, बस अन्तर यह है कि यहाँ चिह्न की बि
को शेष रजिस्टर्स (Registers) से अलग नहीं किया गया है। यहाँ A रजिस्टर को AC [एक्युमुलेटर (Accu
और B रजिस्टर को BR से प्रदर्शित किया गया है। AC और BR की सबसे बायीं ओर वाली बिट (Bit), सं
बिट (Sign Bit) होती है। चिह्न बिट (Sign Bit) को भी अन्य बिट्स की भाँति काम्प्लीमेन्टर (Compleme
एडर (Adder) की सहायता से जोड़ा और घटाया जाता है। यदि ओवरफ्लो (Overflow) होता है, तो फ्लि
(Flip-Flop) V का मान 1 कर दिया जाता है। यहाँ आउटपुट हासिल अर्थात् कैरी (Carry) को हटा दि
साइन्ड 2's काम्प्लीमेन्टस (Signed 2's Complement) में निरूपित दो बायनरी (Binary) संख्याओं को
घटाने की प्रक्रिया के फ्लोचार्ट (Flowchart) को निम्नांकित चित्र 3.4 में प्रदर्शित किया गया है—



चित्र 3.4 Algorithm for Adding and Subtraction Numbers in Signed-2's Complement Representation

Scanned with CamScanner

यहाँ AC और BR के जोड़ और घटा को प्रदर्शित किया गया है। जोड़ को प्राप्त करने के लिए AC और BR को जमा किया जाता है। अन्तिम दो हासिल को एक्सक्लूसिव OR (Exclusive OR) किया जाता है। यदि यह गेट (Gate) 1 उत्पन्न करता है, तो V का मान 1 कर दिया जाता है अन्यथा V का मान 0 कर दिया जाता है। घटा करने के लिए AC को BR के 2's काम्प्लीमेन्ट (2's Complement) के साथ जमा किया जाता है। BR के 2's काम्प्लीमेन्ट (2's Complement) का तात्पर्य है BR को धनात्मक संख्या से ऋणात्मक संख्या में परिवर्तित करना अथवा ऋणात्मक संख्या से धनात्मक संख्या में परिवर्तित करना। चूँकि यदि ओवरफ्लो (Overflow) होता है, तो AC रजिस्टर (register) में गलत उत्तर हो सकता है, अत: यहाँ ओवरफ्लो (Overflow) को जांचना पड़ेगा, क्योंकि हो सकता है कि दो समान चिह्नों वाली संख्याओं को जोड़ा गया है। यहाँ हम देख सकते हैं कि साइन्ड 2's काम्प्लीमेन्ट (Signed 2's Complement) में घटाना, साइन्ड मैग्नीट्यूड (Signed Magnitude) की तुलना में अत्यन्त सरल होता है।

प्रश्न 4. गुणा की एल्गोरिथ्म (Multiplication Algorithm) को समझाइए।

उत्तर **गुणा की एल्गोरिथ्म** Multiplication Algorithm साइन्ड-मैग्नीट्यूड (Signed-Magnitude) निरूपण में दो फिक्स्ड-प्वॉइन्ट बायनरी (Fixed-Point Binary) संख्याओं की गुणा, कागज पर कुछ शिफ्ट (Shift) और जमा की सहायता से किया जा सकता है। इसे समझने के लिए हम एक संख्यात्मक उदाहरण की सहायता लेंगे—

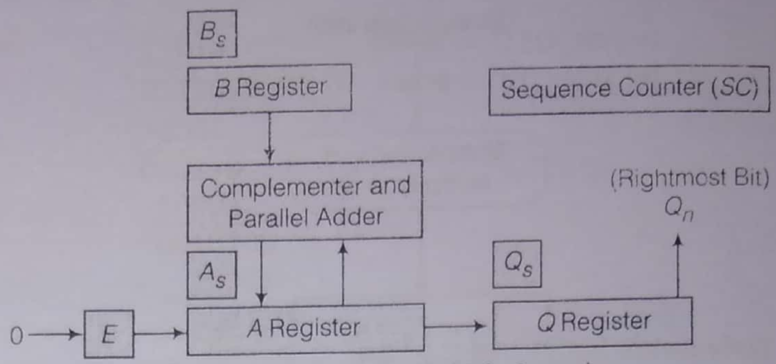
```
 22          10110      गुणज (Multiplicand)
 20        × 10100      गुणक (Multiplier)
 ---       -------
             00000
            00000
           10110
          00000
         10110
        ---------
440     110111000       गुणनफल (Product)
```

यह गुणा, सामान्य गुणा की तरह ही होती है। यहाँ दायीं ओर की बिट से बायीं ओर बढ़ते हुए गुणक की प्रत्येक बिट (Bit) को गुणज से गुणा की जाती है। यदि गुणक बिट (Bit) का मान 1 है तो पूरे गुणक को लिख लिया जाता है, अन्यथा बिट की संख्या के समान 0 को लिखा जाता है। जब दूसरी लाइन लिखी जाती है तो पहली लाइन से बायीं ओर एक अंक विस्थापित (Shift) हो जाता है। अन्ततः समस्त संख्याओं को जोड़ लिया जाता है और यह योग गुणा के समान होता है। गुणा का चिह्न गुणक व गुणज के चिह्न पर निर्भर करता है, यदि गुणक व गुणज के चिह्न समान हैं, तो गुणा का चिह्न धनात्मक होगा और यदि ये चिह्न भिन्न हैं, तो गुणा का चिह्न ऋणात्मक होगा।

प्रश्न 5. डिजिटल कम्प्यूटर में साइन्ड मैग्नीट्यूड डेटा को हार्डवेयर द्वारा किस प्रकार कार्यान्वित किया जाता है? समझाइए।

उत्तर **साइन्ड मैग्नीट्यूड डेटा के लिए हार्डवेयर द्वारा कार्यान्वित करना** Hardware Implementation for Signed Magnitude Data डिजिटल कम्प्यूटर में जब गुणा की जाती है तो क्रिया को थोड़ा परिवर्तित करना ही सुविधाजनक होता है। पहले, उतने रजिस्टर (Register), जितने कि गुणक संख्या में अंक हैं, उपलब्ध कराने के स्थान पर एक समानान्तर एडर (Parallel Adder) को लगा दें और रजिस्टर में रखे डेटा का जोड़ कराकर A रजिस्टर में योग सुरक्षित करें। इस प्रकार आंशिक गुणा को प्राप्त किया जा सकता है। दूसरे, गुणज को बायीं ओर सरकाने के स्थान पर, आंशिक गुणा को दायीं ओर सरकायें और अन्ततः जब गुणक की बिट का मान 0 हो, तो समस्त 0 को लिखने की आवश्यकता नहीं होती, चूँकि 0 को जोड़ने से आंशिक गुणा में कोई परिवर्तन नहीं आयेगा। गुणा के लिए आवश्यक हार्डवेयर को अग्रांकित चित्र 3.5 में दर्शाया गया है—

Scanned with CamScanner



चित्र **3.5** Hardware for Multiply Operation

यहाँ हम देख सकते हैं, कि A और B रजिस्टर के साथ दो और रजिस्टर्स SC और Q भी दिये गये हैं। रजिस्टर Q के अन्दर गुणक को और Q_s के अन्दर गुणक के चिह्न को सुरक्षित किया गया है। सीक्वेन्स काउन्टर (Sequence Counter) SC में प्रारम्भ में गुणक की कुल बिट्स (Bits) की संख्या को सुरक्षित किया जाता है। फिर प्रत्येक आंशिक गुणा के साथ में इसमें से 1 कम कर दिया जाता है। जब SC रजिस्टर का मान शून्य हो जाता है, तो इसका तात्पर्य यह है कि हमें वास्तविक गुणा प्राप्त हो गई है और इस क्रिया को रोक दिया जाता है।

प्रारम्भ में गुणज को B में और गुणक को Q में सुरक्षित किया जाता है। A और B का योग आंशिक गुण होता है और उसे EA रजिस्टर में प्रवाहित किया जाता है। आंशिक गुणा और गुणक दोनों को दायीं ओर शिफ्ट (Shift) कर दिया जाता है। A की सबसे बायीं ओर वाली बिट (Bit) को Q की सबसे दायीं ओर वाली बिट (Bit) में, E से बिट (Bit) को A की सबसे दायीं ओर वाली बिट (Bit) में और E में 0 को शिफ्ट (Shift) किया जाता है। शिफ्ट (Shift) के बाद, आंशिक गुणा की एक बिट को Q में शिफ्ट (Shift) किया जाता है, इससे गुणक की बिट्स (Bits) एक स्थान दायीं ओर सरक जाती है। इस प्रकार Q रजिस्टर के सबसे दायें वाले फ्लिप-फ्लाप अर्थात् Q_n में अगली जांची जाने वाली गुणक की बिट होगी।

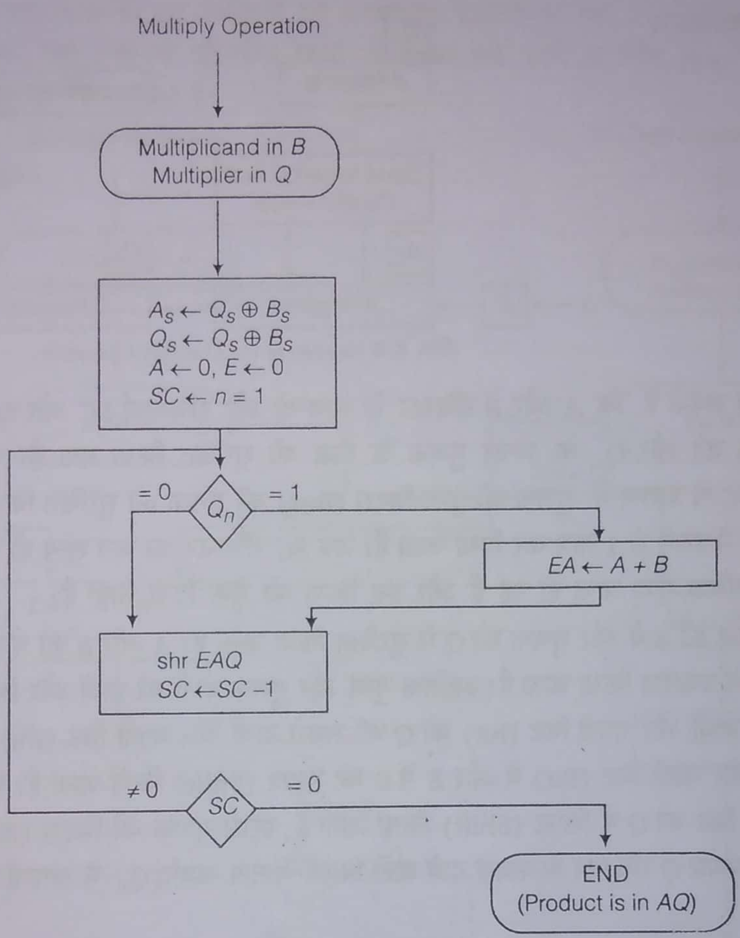
प्रश्न 6. गुणा संक्रिया की हार्डवेयर एल्गोरिथ्म (Hardware Algorithm) को समझाइए।

उत्तर **हार्डवेयर एल्गोरिथ्म** Hardware Algorithm चित्र 3.6 में एक फ्लोचार्ट (Flowchart) दिया गया है जिसमे गुणा की एल्गोरिथ्म (Algorithm) को प्रदर्शित किया गया है—

प्रारम्भ में B में गुणज और Q में गुणक होता है और इनके चिह्न क्रमश: B_s और Q_s हैं। इन दोनों चिह्नों की तुलना की जाती है और इनके अनुसार ही A और Q को सेट (Set) किया जाता है। रजिस्टर A और E में 0 रख दिया जाता है और SC के अन्दर गुणक की बिट्स की संख्या के समान अंक को सुरक्षित किया जाता है।

इसके पश्चात्, Q_n में सुरक्षित निचले स्तर की बिट (Bit) को जांचा जाता है। यदि इसका मान 1 है, तो A में सुरक्षित आंशिक गुणा को B में सुरक्षित गुणज के साथ जोड़ा जाता है। यदि इसका मान 0 है, तो कुछ नहीं किया जाता है। नया आंशिक गुणा बनाने के लिए रजिस्टर EAQ को दायीं ओर एक बार शिफ्ट (Shift) किया जाता है। फिर SC के मान में 1 की कमी की जाती है और SC के मान को जांचा जाता है। यदि यह 0 नहीं है तो क्रिया की पुनरावृत्ति की जाती है और नया आंशिक गुणा प्राप्त करते हैं। यदि यह 0 है तो क्रिया को रोक दिया जाता है। ध्यान दें कि अन्तिम गुणा A और Q दोनों में सुरक्षित है, जहाँ A में दायीं ओर वाली बिट्स और B में बायीं ओर की बिट्स होती है। अत: पूर्ण गुणा हमें AQ से प्राप्त होगी।

Scanned with CamScanner



चित्र **3.6** Flowchart for Multiply Operation

प्रश्न 7. साइन्ड (Signed) 2's काम्प्लीमेन्ट के साथ गुणा एल्गोरिथ्म को समझाइए।

उत्तर **साइन्ड** 2's **काम्प्लीमेन्ट के साथ गुणा अथवा बूथ गुणा एल्गोरिथ्म** Multiplication with Signed 2's Complement or Booth Multiplication Algorithm बूथ एल्गोरिथ्म (Booth Algorithm), साइन्ड 2's काम्प्लीमेन्ट (Signed 2's Complement) द्वारा निरूपित बायनरी अंकों की गुणा करने के लिए प्रयोग किया जाता है। यह इस तथ्य पर कार्य करता है कि, यदि गुणक में 0 है, तो कोई जोड़ नहीं करना होगा, बल्कि केवल शिफ्ट (Shift) करना होगा, और यदि गुणक में 1 है, तो उसमें सबसे बायीं ओर के 1 की 2 की घात (k) से लेकर, सबसे दायीं ओर के 1 की 2 की घात (m) तक के शब्द के साथ $2^{k+1} - 2^m$ की भाँति व्यवहार किया जाता है। उदाहरण के लिए, बायनरी अंक 001110 (दशमलव अंक + 14) में 1 की स्ट्रिंग (String) 2^3 से 2^1 तक है जहाँ $k = 3$ और $m = 1$ है। अत: इसे $2^{k+1} - 2^m$ के अनुसार $2^4 - 2^1 = 16 - 2 = 14$ की भाँति प्रदर्शित किया जा सकता है।

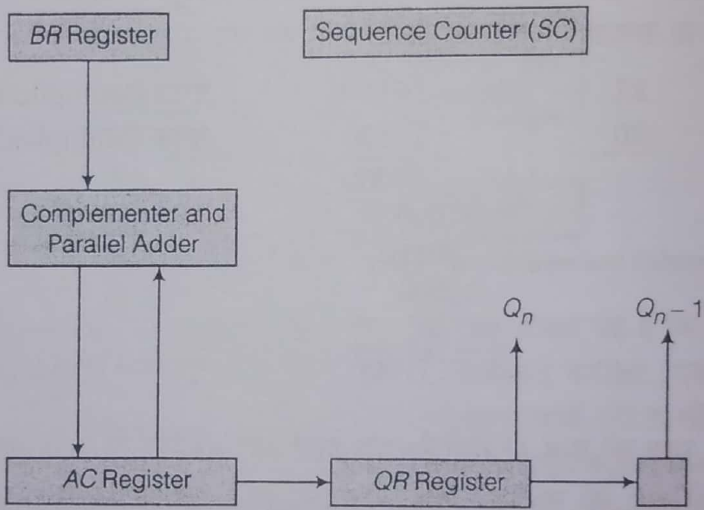
अत: गुणा $M \times 14$ है, जहाँ M गुणज और 14 गुणक है और इस गुणा को $M \times 2^4 - M \times 2^1$ के रूप में किया जा सकता है। अत: गुणा को करने के लिए पहले गुणज M को चार बार बायीं ओर शिफ्ट (Shift) किया जायेगा और इसमें से एक बार बायीं ओर शिफ्ट (Shift) को घटाया जायेगा।

गुणा की सभी एल्गोरिथ्म (Algorithm) के अनुसार, बूथ एल्गोरिथ्म (Algorithm) में भी गुणक को जांचना पड़ता है और आंशिक गुणा को शिफ्ट (Shift) करना होता है। आंशिक गुणा में निम्नलिखित परिवर्तन शिफ्ट (Shift) के पहले करने पड़ते हैं—

Scanned with CamScanner

1. गुणक के 1 वाली स्ट्रिंग (String) में सबसे बायीं ओर वाले 1 के जांचे जाने पर आंशिक गुणा में से गुण घटाया जाता है।
2. गुणक के 0 वाली स्ट्रिंग (String) में पहला 0 (यह माना जाता है कि इससे पहले 1 होगा) जांचे जाने पर अ गुणा में गुणज को जोड़ा जाता है।
3. यदि गुणक बिट पिछली गुणक बिट के समान है, तो आंशिक गुणा में कोई परिवर्तन नहीं होता है।

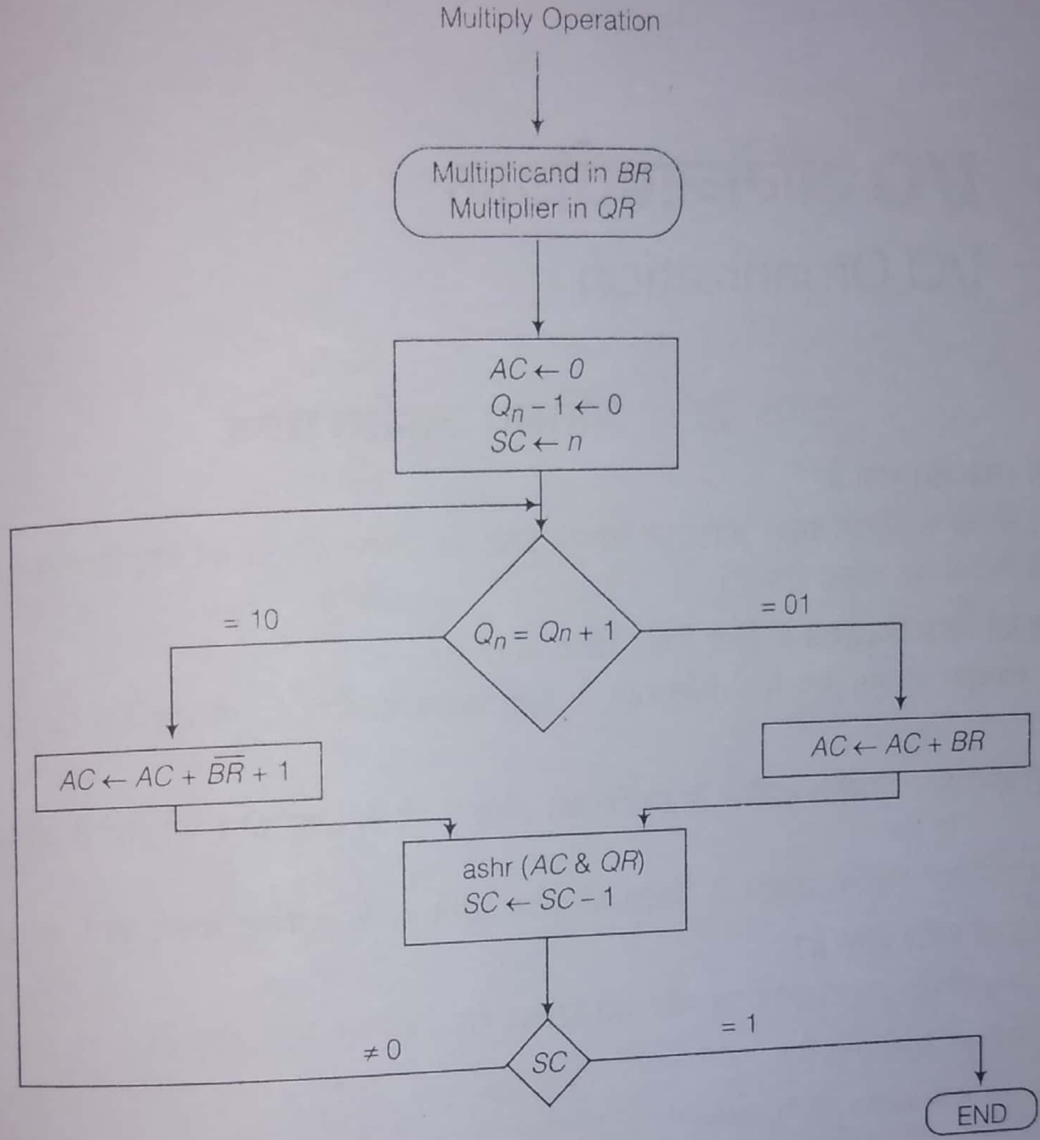
यह एल्गोरिथ्म (Algorithm) 2's काम्प्लीमेन्ट (2's Complement) में निरूपित धनात्मक अथवा ऋणात्मक अं कार्य करता है। यह इसलिए होता है, क्योंकि ऋणात्मक अंक 1 पर समाप्त होता है और अन्तिम अ (Operation) 2^1 घटाने का होता है। उदाहरण के लिए, – 14 को 2's काम्प्लीमेन्ट (2's Complement) के 110010 के रूप में निरूपित किया जाता है और $-2^4 + 2^2 - 2^1 = -14$ की भाँति व्यवहार करता है। बूथ एल्गोरिथ्म (Booth Algorithm) को हार्डवेयर के साथ कार्यान्वित करने के लिए, निम्नांकित रजिस्टर कॉनफि (Register Configuration) की आवश्यकता होती है।



चित्र **3.7** Hardware for Booth Algorithm

यह साइन्ड मैग्नीट्यूट डेटा (Signed Magnitude Data) के लिए गुणा की भाँति ही है, अन्तर केवल यही है कि चिह्न की बिट (Sign Bit) को प्रदर्शित नहीं किया है, ऐसा इसलिये किया गया है क्योंकि 2's कॉप्लीमेन्ट Complement) में चिह्न को शब्द में ही रखा जाता है। यहाँ *A*, *B* और *Q* रजिस्टर को *AC*, *BR* और *QR* के प्रदर्शित किया है। *QR* रजिस्टर की सबसे बायीं ओर वाली बिट को Q_n से प्रदर्शित किया गया है। एक फ्लिप (Flip-Flop) Q_{n+1} को *QR* के साथ जोड़ा गया है, जिससे गुणक की बिट (Bit) को दो बार जांचा जा सकता एल्गोरिथ्म (Booth Algorithm) को चित्र 3.8 में प्रदर्शित किया गया है।

यहाँ हम देख सकते हैं कि प्रारम्भ में *AC* और Q_{n+1} में 0 को रखा गया है और सीक्वेन्स काउन्टर (Seq Counter) *SC* में गुणक की कुल बिट्स (Bits) की संख्या *n* को सुरक्षित किया गया है। तत्पश्चात् गुणक की दो (Bits) Qn और Q_{n+1} जांचा जाता है। यदि इसका मान 10 है, तो इसका तात्पर्य है कि 1's की स्ट्रिंग (Stri पहले 1 को जांचा गया है, उपरोक्त फ्लोचार्ट में दिये गये चरणों के अनुसार, इसका तात्पर्य है कि *AC* में आंशिक गुणा में से गुणज को घटाया जायेगा। यदि Qn और Q_{n+1} का मान 01 है, तो इसका तात्पर्य है कि यह स्ट्रिंग (String) के पहले 0 को जांचा गया है और उपरोक्त फ्लोचार्ट में दिये गये चरणों के अनुसार इसका ता



चित्र **3.8** Booth Algorithm for Multiplication for Signed-2's Complement

कि गुणज को आंशिक गुणा में जोड़ा जायेगा। यदि Qn का मान Q_{n+1} के बराबर है, तो इसका तात्पर्य है कि आंशिक गुणा में कोई परिवर्तन नहीं होगा। अगले चरण में AC और QR अर्थात् आंशिक गुणा और गुणक को दायीं ओर शिफ्ट (Shift) किया जायेगा। यह ऑपरेशन अर्थमेटिक शिफ्ट राइट (Arithmetic Shift Right-ashr) है, जो AC और QR को दायीं ओर शिफ्ट (Shift) करेगा और AC की चिह्न बिट (Sign bit) को परिवर्तन नहीं करेगा। तत्पश्चात् सीक्वेन्स काउन्टर (Sequence Counter) SC में से 1 को घटा दिया जाता है। अन्ततः SC के मान को जांचा जाता है। यदि यह 0 नहीं है, तो Qn और Q_{n+1} के मान को जांचा जायेगा और पूरी क्रिया को दोहराया जायेगा और यदि SC का मान 0 है, तो AC में अन्तिम गुणा है। अतः यह पूरी क्रिया तब तक दोहरायी जायेगी जब तक SC का मान 0 नहीं हो जायेगा।

प्रश्न 8. भाग (division) की एल्गोरिथ्म को लिखिए।

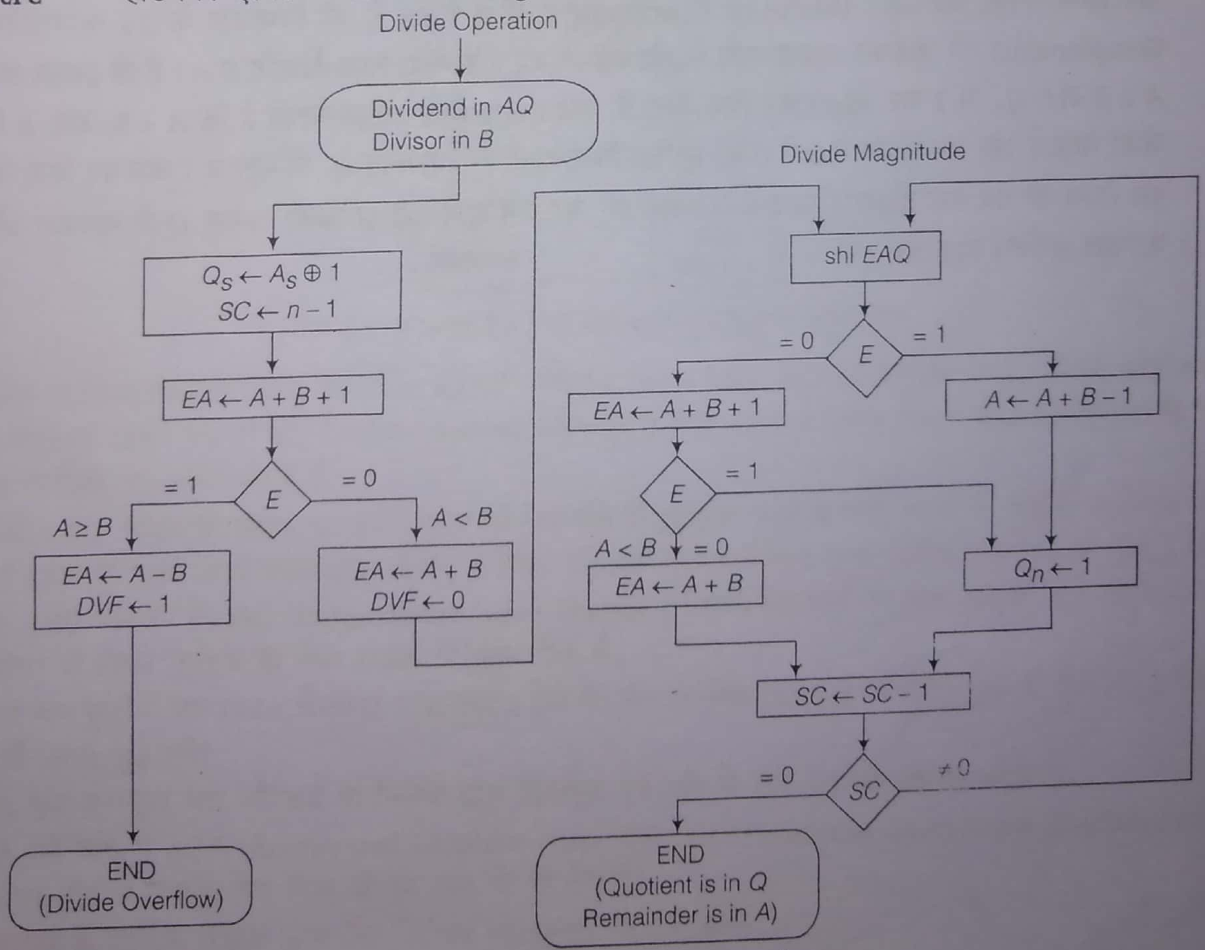
उत्तर **भाग की एल्गोरिथ्म** Division Algorithm साइन्ड मैग्नीट्यूड (Signed Magnitude) द्वारा निरूपित दो फिक्स्ड पॉइन्ट बायनरी (Fixed Point Binary) संख्याओं के भाग को कागज पर पिछले उत्तर से तुलना, घटा और शिफ्ट (Shift) के द्वारा किया जा सकता है। इसे समझने के लिए हम एक संख्यात्मक उदाहरण की सहायता लेंगे—

```
विभाजक (Divisor)          11010        भागफल = Q (Quotient)
B = 10001            )0111000000         भाज्य = A (Dividend)
                       10001
                       010110
                        10001
                        0010100
                          10001
                          000110
                           00110      अन्तिम शेषफल (Final Remainder)
```

यहाँ हम देख सकते हैं कि यह बहुत सरल है। पहले भाग्य की 5 बिट् (Bits) की विभाजक की 5 बिट्स (Bits) से तुलना करेंगे। उदाहरण में हम देख सकते हैं, कि भाज्य की 5 बिट्स (Bit) विभाजक की 5 बिट्स (Bit) से कम हैं। अत: भाज्य की 6 बिट (Bit) लेंगे और भागफल में 1 लिखेंगे और भाज्य में से विभाजक को घटायेंगे। घटाने के बाद प्राप्त शेषफल को आंशिक शेषफल कहा जाता है। इस प्रकार पूरी प्रक्रिया को तब तक दोहराया जाता है जब तक अभाज्य शेषफल प्राप्त नहीं हो जाता। इस भाग के उत्तर में भागफल और शेषफल प्राप्त होता है।

प्रश्न 9. भाग विधि की हार्डवेयर एल्गोरिथ्म विधि का फ्लोचार्ट सहित वर्णन कीजिए।

उत्तर हार्डवेयर एल्गोरिथ्म Hardware Algorithm को निम्नांकित फ्लोचार्ट में समझाया गया है—



चित्र **3.9** Flowchart for Divide Operation

Scanned with CamScanner

A और Q में भाज्य तथा B में विभाजक को सुरक्षित किया गया है। उत्तर के चिह्न को Q_s द्वारा निरूपित किया गया है। सीक्वेन्स काउन्टर (Sequence Counter) SC को भागफल की सम्भावित बिट्स (Bits) की संख्या से इनिशियलाइज (Initialize) किया जाता है। जैसा कि गुणा में माना जाता है कि ऑपरेन्ड्स (Operands) को n बिट्स (Bits) के शब्दों वाली मेमोरी (Memory) में से रजिस्टर (Register) में भेजा जाता है। चूँकि एक ऑपरेन्ड (Operand) को उसके चिह्न के साथ सुरक्षित करना होता है, अत: चिह्न 1 बिट (Bit) लेगा और मैग्नीट्यूड (Magnitude) $n-1$ बिट्स (Bits) में सुरक्षित होगा।

भाग-ओवरफ्लो (Divide-Overflow) स्थिति को जांचने के लिए A में सुरक्षित भाज्य की आधी बिट्स (Bits) में से B को घटाया जाता है। यदि $A \geq B$ है, तो डिवाइड ओवरफ्लो फ्लिप-फ्लाप (Divide Overflow Flip-Flop) को सेट (Set) कर दिया जाता है और भाग के कार्य को रोक दिया जाता है। यदि $A < B$ है भाग-ओवरफ्लो (Divide-Overflow) नहीं होगा। इसलिए A के मान में विभाजक B के मान को जोड़कर भाज्य का मान पुनर्निर्धारित किया जाता है।

मैग्नीट्यूड (Magnitude) के भाग को प्रारम्भ करने के लिए AQ में बायीं ओर से भाज्य को डालें और E में उच्च स्तर की बिट (Bit) को सुरक्षित करें। यदि E में शिफ्ट (Shift) की गई बिट (Bit) का मान 1 हो, तो हम जानते हैं कि $EA > B$, क्योंकि अब E में 1 है और A में $n-1$ बिट्स (Bits) है, अत: EA में कुल n बिट्स (Bits), जबकि B में कुल $n-1$ बिट्स (Bits) हैं। इस स्थिति में EA में से B को घटाना चाहिये और भागफल बिट के लिए Q_n में 1 को भेजें। यदि शिफ्ट-लेफ्ट ऑपरेशन (Shift-Left Operation) E में 0 भेजता है, तो विभाजक के 2's काम्प्लीमेन्ट (2's Complement) को जोड़कर घटाया जाता है और हासिल को E में भेजा जाता है। यदि $E = 1$ है तो इसका तात्पर्य है, $A \geq B$ अत: Q_n में 1 सेट (Set) कर दिया जाता है। यदि $E = 0$ है तो इसका तात्पर्य है कि $A < B$, अत: B में A को जोड़ा जाता है और इस स्थिति में Q में 0 को सुरक्षित किया जाता है। तत्पश्चात् SC के मान से 1 कम कर दिया जाता है। इस क्रिया को तब तक दोहराया जाता है जब तक SC का मान शून्य नहीं हो जाता। अन्तत: Q में भागफल और A में शेषफल सुरक्षित होता है।

Scanned with CamScanner

I/O ऑर्गेनाइजेशन
I/O Organization

4

खण्ड 'अ' : अतिलघु उत्तरीय प्रश्न

प्रश्न 1. बायोस (BIOS) क्या है?

उत्तर बायोस अंग्रेजी के बेसिक इनपुट आउटपुट सिस्टम शब्द का संक्षिप्त रूप है। यह आईबीएम (IBM) कम्प्यूटर को दिए जाने वाले निर्देशों का समूह होता है।

प्रश्न 2. बायोस को किस उद्देश्य के लिए तैयार किया जाता है?

उत्तर निर्देश कम्प्यूटर में एक चिप में संरक्षित रहते हैं। अत: डिस्क को फेल होने से बचाने के लिए बायोस को तैयार किया जाता है।

प्रश्न 3. पर्सनल कम्प्यूटर निर्माता बायोस के निर्देश को लोड करने के लिए कौन-सी मेमोरी का इस्तेमाल करते हैं?

उत्तर पर्सनल कम्प्यूटर निर्माता बायोस के निर्देश को लोड करने के लिए फ्लैश मेमोरी कार्ड का प्रयोग करते हैं।

प्रश्न 4. BIOS कहाँ स्टोर होता है?

उत्तर BIOS का सॉफ्टवेयर मदरबोर्ड पर नॉन-व्होलटाइल रोम चिप पर स्टोर होता है।

प्रश्न 5. DMA का पूरा नाम क्या है?

उत्तर DMA—Direct Memory Access.

खण्ड 'ब' : लघु एवं दीर्घ उत्तरीय प्रश्न

प्रश्न 1. बायोस (BIOS) पर संक्षिप्त टिप्पणी लिखिए।

उत्तर कम्प्यूटर चलाते समय सर्वप्रथम स्क्रीन पर हमें बायोस दिखायी पड़ता है। BIOS का पूरा नाम Basic Input/Output System है। इसे सिस्टम बायोस या PC बायोस भी कहा जाता है। यह मदरबोर्ड के साथ जुड़ा एक सॉफ्टवेयर होता है जो कम्प्यूटर स्टार्ट करते ही स्वयं चालू हो जाता है। बायोस कम्प्यूटर के चालू होने पर RAM, प्रोसेसर, की-बोर्ड, हार्ड-डिस्क आदि की पहचान कर उनका वर्गीकरण करता हैं। यह कम्प्यूटर की रोम मेमोरी में स्थित होता है जो कम्प्यूटर के सभी हार्डवेयर को पहचानता है। बायोस मुख्य रूप से कम्प्यूटर के ऑपरेटिंग सिस्टम (Operating System) को बूट कराने का कार्य करता है। जब कम्प्यूटर ऑन होता है तो BIOS सीमोस सेटअप की जाँच करता है और यह तय करता है कि किस डिवाइस से सिस्टम को बूट किया जाए एवं इन सभी को तैयार (Prepare) करने के बाद ऑपरेटिंग सिस्टम को मैसेज करता है, इसके बाद ही कम्प्यूटर कार्य करने के लिए तैयार होता है। BIOS प्रत्येक कम्प्यूटर में पहले से भी इनस्टॉल्ड रहता है क्योंकि CPU में प्रोग्राम BIOS के बाद ही इनस्टॉल होते हैं यह ROM के अन्दर इनस्टॉल रहता है एवं यह चिप मदरबोर्ड में लगी होती है।

प्रश्न 2. कम्प्यूटर सिस्टम में बायोस के कार्य लिखिए।

उत्तर कम्प्यूटर सिस्टम में BIOS एक सिक्योरिटी गार्ड की तरह से कार्य करता है। यह कम्प्यूटर सिस्टम के अंदर जाने से पहले पूरी जानकारी लेता है कि बेसिक फीचर्स कम्प्यूटर सिस्टम में लगे हुए हैं या नहीं। अगर ये सारे फीचर्स कम्प्यूटर सिस्टम में नहीं हैं तो कम्प्यूटर चालू नहीं होगा।

BIOS के मुख्य कार्य निम्नलिखित हैं—

(i) यह पीसी पर ऑपरेटिंग सिस्टम को बूट करता है।

Scanned with CamScanner

(ii) कस्टम सेटिंग के लिए सीमोस सेटअप की जाँच करता है।

(iii) Power-onself Test (POST) को प्रदर्शित करता है।

(iv) BIOS हमारे CPU के सारे रजिस्टरों को ठीक तरह से स्थापित अर्थात् उपयोग होने के लिए तैयार करता है।

(v) BIOS कम्प्यूटर ड्राइवरर्स को लोड करता है जो ऑपरेटिंग सिस्टम तथा कनेक्टिड डिवाइस के बीच एक बेस का कार्य करते हैं।

(vi) BIOS सेटिंग को लोड करता है।

(vii) BIOS एक बूटेबल माध्यम ढूँढता है और उसके बाद बूटेबल माध्यम को रीड करके जरूरी फाइल्स को RAM में लोड करता है एवं इसके बाद कम्प्यूटर ऑन होता है तथा डेस्कटॉप दिखाई देता है।

प्रश्न 3. डाटा ट्रांसफर विधियों का वर्णन कीजिए।

उत्तर डाटा ट्रांसफर करने की निम्नलिखित तीन विधियाँ हैं—

1. **सिंक्रोनस डाटा ट्रांसफर विधि** डाटा ट्रांसफर की यह सबसे आसान विधि है। यह विधि तब प्रयोग की जाती है जब I/O डिवाइस और माइक्रोप्रोसेसर की गति एकसमान होती है। डिवाइस से डाटा ट्रांसफर करने के लिए डिवाइस को उचित इंटरसेक्शन दी जाती है।

   डाटा ट्रांसफर में पेरिफेरल मैपड (Mapped) I/O डिवाइस के साथ इन (IN) या आउट (Out) इंटरसेक्शन और मेमोरी मैपड I/O डिवाइस के साथ मेमोरी रीड/राइट इंटरसेक्शन का उपयोग किया जाता है। इंटरसेक्शन इन (IN) का प्रयोग इनपुट डिवाइस या इनपुट पोर्ट से डाटा 'रीड' करने के लिए किया जाता है। इंटरसेक्शन आउट (Out) का प्रयोग CPU से आउटपुट डिवाइस या आउटपुट पोर्ट पर डाटा भेजने के लिए किया जाता है। अब CPU तथा I/O डिवाइस डाटा ट्रांसफर के लिए तैयार रहती हैं। इस प्रक्रिया में I/O डिवाइस का स्टेट्स जाँचने की जरूरत नहीं होती है।

2. **असिंक्रोनस डाटा ट्रांसफर विधि** जब I/O डिवाइस और माइक्रोप्रोसेसर की गति परस्पर एकसमान नहीं होती है तो असिंक्रोनस डाटा ट्रांसफर विधि का प्रयोग किया जाता है। इस विधि में डाटा ट्रांसफर करने से पहले माइक्रोप्रोसेसर I/O डिवाइस का स्टेट्स चैक किया जाता है कि डिवाइस डाटा ट्रांसफर के लिए तैयार है या नहीं। माइक्रोप्रोसेसर I/O डिवाइस को पहले तैयार होने के लिए सिगनल भेजता है तथा I/O डिवाइस का स्टेट्स लगातार चेक करता रहता है जब तक कि I/O डिवाइस डाटा ट्रांसफर के लिए तैयार नहीं हो जाती। जब I/O डिवाइस तैयार हो जाती है तब माइक्रोप्रोसेसर डाटा ट्रांसफर निर्देश भेजता है। इस विधि को हैंडसेकिंग भी कहा जाता है; क्योंकि डाटा ट्रांसफर से पहले माइक्रोप्रोसेसर तथा I/O डिवाइस के मध्य सिगनल का एक्सचेंज होता है। असिंक्रोनस डाटा ट्रांसफर धीमी तकनीकी विधि I/O डिवाइस के साथ प्रयुक्त की जाती है। इस विधि की दक्षता कम है; क्योंकि माइक्रोप्रोसेसर का काफी समय इंतजार में नष्ट होता है।

3. **इंटरप्ट ड्राइव डाटा ट्रांसफर विधि** इंटरप्ट विधि द्वारा डाटा ट्रांसफर उस स्थिति में किया जाता है जब I/O डिवाइस की गति माइक्रोप्रोसेसर की गति से कम होती है। उदाहरणार्थ—करेक्टर प्रिंटर, AD कनवर्टर आदि। इंटरप्ट डाटा ट्रांसफर में माइक्रोप्रोसेसर सबसे पहले I/O डिवाइस को तैयार होने का सिगनल देता है तथा अपने में प्रोग्राम को एक्जीक्यूट कर देता है। जब I/O डिवाइस डाटा ट्रांसफर के लिए तैयार हो जाती है तब ये इंटरप्ट लाइन द्वारा माइक्रोप्रोसेसर को उच्च सिगनल देती है अर्थात् ये माइक्रोप्रोसेसर की सामान्य क्रिया को बाधित करती है। इंटरप्ट सिगनल को प्राप्त करने के बाद माइक्रोप्रोसेसर उस समय एक्जीक्यूट हो रहे निर्देश को पूरा करता है तब I/O डिवाइस पर ध्यान केन्द्रित करता है। सबसे पहले यह प्रोग्राम काउन्टर के कन्टेन्ट स्टैक में सेव करता है तथा उसके बाद एक सबरुटीन में जाता है जिसे इंटरप्ट सर्विस सबरुटीन या ISS कहते हैं। माइक्रोप्रोसेसर I/O डिवाइस से डाटा रीड करने के लिए या राइट करने के लिए ISS को एक्जीक्यूट करता है। डाटा ट्रांसफर की प्रक्रिया पूरी होने पर माइक्रोप्रोसेसर फिर से उसी प्रोग्राम में वापस आ जाता है जिसे वह इंटरप्ट होने से पहले एक्जीक्यूट कर रहा था।

Scanned with CamScanner

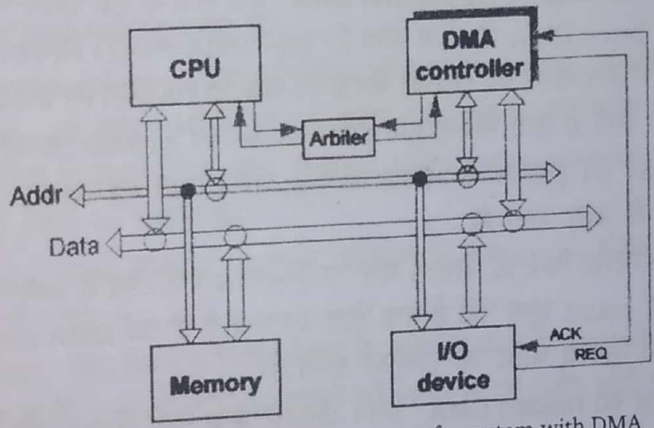
इंटरप्ट डाटा ट्रांसफर विधियों, असिंक्रोनस डाटा ट्रांसफर से अधिक कुशल है, क्योंकि इसमें माइक्रोप्रोसेसर का वह कीमती समय नष्ट नहीं होता जो वह I/O डिवाइस के तैयार होने में व्यतीत होता है।

प्रश्न 4. DMA क्या है? DMA द्वारा डाटा ट्रांसफर करने की विधियों का वर्णन कीजिए।

उत्तर DMA अर्थात् (Direct Memory Access) डायरेक्ट मेमोरी एक्सेस है। DMA एक ऐसी विधि है जिसके द्वारा डाटा को कम्प्यूटर की RAM से इनपुट/आउटपुट डिवाइस में बिना सीपीयू का प्रयोग किये ट्रांसफर किया जाता है। इससे मेमोरी ऑपरेशनों की गति बढ़ती है तथा डेटा ट्रांसफर की गति भी बढ़ती है। डेटा ट्रांसफर में वही समय लगता है जो मेमोरी से डाटा रीड तथा राइट करने में लगता है इसमें डाटा एक ब्लॉक के रूप में ट्रांसफर होता है।

कम्प्यूटर में अधिकतर डाटा सीपीयू के द्वारा प्रोसेस होता है किन्तु कम्प्यूटर में कुछ डाटा को प्रोसेस करने की जरूरत नहीं होती है इस स्थिति में DMA जो है वह प्रोसेसिंग में लगने वाले समय को सेव कर सकता है तथा डेटा को अच्छे तरीके से कम्प्यूटर की मेमोरी से दूसरे डिवाइसों में ट्रांसफर कर सकता है।

यह पूरी प्रक्रिया एक चिप के द्वारा मैनेज होती है जिसे हम DMA कंट्रोलर (DMAC) कहते हैं। DMA कंट्रोलर डाटा को contain नहीं करता है बल्कि एक केवल बस को ड्राइव करता है। DMA कंट्रोलर एक I/O डिवाइस को DMA सिग्नल द्वारा एक्सेस करता है।



चित्र **4.1** Simplified logical structure of a system with DMA

अनेक हार्डवेयर सिस्टम DMA का प्रयोग करते हैं, जैसे कि डिस्क ड्राइव कंट्रोलर्स ग्राफिक कार्ड, नेटवर्क कार्ड तथा साउंड कार्ड आदि। जब भी कोई डिवाइस DMA का प्रयोग करती है तो उसे एक विशेष DMA चैनल को विशिष्ट निर्देश देता है जिसे वह प्रयोग करती है।

अल्ट्रा DMA ड्राइव जो DMA का प्रयोग डाटा को तेज गति से ट्रांसफर करने के लिए करती है, जबकि पहले कि जो हार्ड ड्राइव थी उसमें सबसे पहले डाटा सीपीयू से होकर ट्रांसफर होता था जिसमें समय अधिक लग जाता था। DMA का एक दूसरा विकल्प है, PIO (Programmed Input/Output), जिसमें डिवाइसों के मध्य ट्रांसफर होने वाला डाटा प्रोसेसर से होकर गुजरता है। PIO DMA से धीमा होता है।

पुराने कम्प्यूटरों में चार DMA चैनल 0, 1, 2 तथा 3 होते थे। जब 16 बिट ISA bus प्रस्तावित हुआ तो चैनल 5, 6 तथा 7 भी उसमें जुड़ गये।

ISA एक कम्प्यूटर बस स्टैण्डर्ड था जिसके द्वारा डिवाइस तेज गति से लेन-देन को आरम्भ करते थे। ISA को बाद में AGP (Accelerated Graphics Port) तथा PCI (Peripheral Component Interconnect) विस्तार बोर्ड ने रिपलेस कर दिया जो कि ISA से भी तेज है।

कम्प्यूटर के सिस्टम रिसोर्स टूल्स निम्नलिखित चार प्रकार के होते हैं—

1. I/O addresses
2. Memory address

Scanned with CamScanner

3. Interrupt request number (IRQ)

4. DMA channels

DMA चैनलों का प्रयोग सिस्टम मेमोरी तथा पेरिफेरल डिवाइसों के मध्य डेटा को कम्प्यूनिकेट करने के लिए किया जाता है। ये सभी चारों सिस्टम रिसोर्स जो है वह बस की किसी नियत लाइन पर निर्भर रहती है। बस में कुछ लाइन्स IRQs के लिए प्रयोग होती है एवं कुछ address के लिए तथा कुछ DMA चैनलों के लिए।

DMA Transfer Modes निम्नलिखित हैं—

1. **burst या block transfer DMA** यह सबसे तेज DMA मोड है इसमें दो या दो से अधिक डाटा bytes को लगातार ट्रांसफर किया जा सकता है। इसमें DMA जो है वह सिस्टम बस के लिए प्रोसेसर को HOLD सिगनल भेजता है तथा HLDA सिगनल का इन्तजार करता है। HLDA सिगनल जब उसे रिसीव हो जाता है तो उसके बाद DMA सिस्टम बस पर नियन्त्रण कर लेता है तथा एक byte भेजता है। एक byte भेजने के बाद यह मेमोरी एड्रेस को बढ़ा देता है एवं काउंटर को घटा देता है तथा फिर दूसरी byte को भेजता है। इस प्रकार यह सभी डाटा bytes को मेमोरी तथा I/O डिवाइस के मध्य ट्रांसफर कर लेता है। सभी डाटा bytes को ट्रांसफर करने के बाद DMA कंट्रोलर जो है वह HOLD सिगनल को disable कर देता है तथा salve मोड में चला जाता है।
2. **cycle steal या single byte transfer DMA** इस मोड में एक समय में केवल एक बाइट ही ट्रांसफर की जाती है। यह मोड burst DMA से धीमा होता है। इसमें DMA HOLD सिगनल को प्रोसेसर को भेजता है तथा HLDA सिगनल को रिसीव करने का इन्तजार करता है। जब HLDA सिगनल इसे प्राप्त हो जाता है तो यह सिस्टम बस को नियंत्रित कर लेता है तथा केवल एक DMA cycle को execute करता है।

   जब एक बाइट को ट्रांसफर कर लिया जाता है, उसके बाद यह HOLD सिगनल को disable कर देता है तथा salve मोड में चला जाता है।

   इसके बाद प्रोसेसर फिर से सिस्टम बस पर नियन्त्रण कर लेता है तथा अगली machine cycle को execute करता है यदि count शून्य नहीं है तथा डेटा उपलब्ध है तो तब DMA controllers प्रोसेसर को HOLD सिगनल भेजते हैं तथा अगली बाइट को ट्रांसफर करते हैं।
3. **transparent या hidden DMA** इसमें प्रोसेसर कुछ ऐसी status को execute करता है जिससे वह सिस्टम बस से अलग जाता है DMA जो है वह इन status के दौरान मेमोरी तथा I/O डिवाइसों के मध्य डेटा ट्रांसफर करता है और यह ऑपरेशन, प्रोसेसर के लिए transparent रहती है। यह सबसे धीमा डाटा ट्रांसफर मोड है।

Scanned with CamScanner

5 8085 माइक्रोप्रोसेसर का परिचय
Introduction to 8085 Microprocessor

खण्ड 'अ' : अतिलघु उत्तरीय प्रश्न

प्रश्न 1. 16-बिट प्रोसेसर की मेमोरी को एड्रैस भेजने की क्षमता कितनी होती है?

उत्तर 1 M बाइट।

प्रश्न 2. 8085 माइक्रोप्रोसेसर में TRAP Interrupt का Branch Address क्या होता है?

उत्तर 0024 CH.

प्रश्न 3. Intel 8085 किस सन् में निर्मित किया गया था?

उत्तर सन् 1976 में।

खण्ड 'ब' : लघु एवं दीर्घ उत्तरीय प्रश्न

प्रश्न 1. माइक्रोप्रोसेसर 8085 से आप क्या समझते हैं? चित्र बनाकर दर्शाइए।

उत्तर माइक्रोप्रोसेसर का मुख्य कार्य डाटा ट्रांसफर ऑपरेशन है। डाटा ट्रांसफर प्राय: माइक्रोप्रोसेसर तथा पैरीफे युक्तियों (मेमोरी एवं I/O युक्तियों) के मध्य होता है। डाटा ट्रांसफर को नियन्त्रित करने के लिए माइक्रोप्रोसेसर, कन्ट्रो सिगनल जेनरेट करता है जिसके द्वारा Read एवं Write ऑपरेशन सम्पन्न किये जाते हैं।

माइक्रोप्रोसेसर में समस्त ऑपरेशन्स के उचित टाइमिंग (Timing) तथा कन्ट्रोल के लिए उसमें एक दोलि (Oscillator) होता है जिसे क्लॉक (Clock) कहते हैं। 8085 माइक्रोप्रोसेसर 3MHz क्लॉक फ्रीक्वेंसी पर प्रचालि होता है। इसकी प्रचालन वोल्टेज + 5 V d.c. है। माइक्रोप्रोसेसर 8085 एक 8-बिट माइक्रोप्रोससर है। यह 64 K मेमो रजिस्टर्स एड्रैस कर सकता है। 8085 की चिप में 40 पिन होती हैं।

चित्र 5.1 में माइक्रोप्रोसेसर-8085 की आन्तरिक संरचना (Internal Structure) प्रदर्शित की गयी हैं इसके मुख्य भा निम्न हैं—

(i) Arithmetic and Logic Unit (ALU)

(ii) टाइमिंग एवं कन्ट्रोल यूनिट (Timing and Control Unit)

(iii) इन्सट्रक्शन रजिस्टर (Instruction Register)

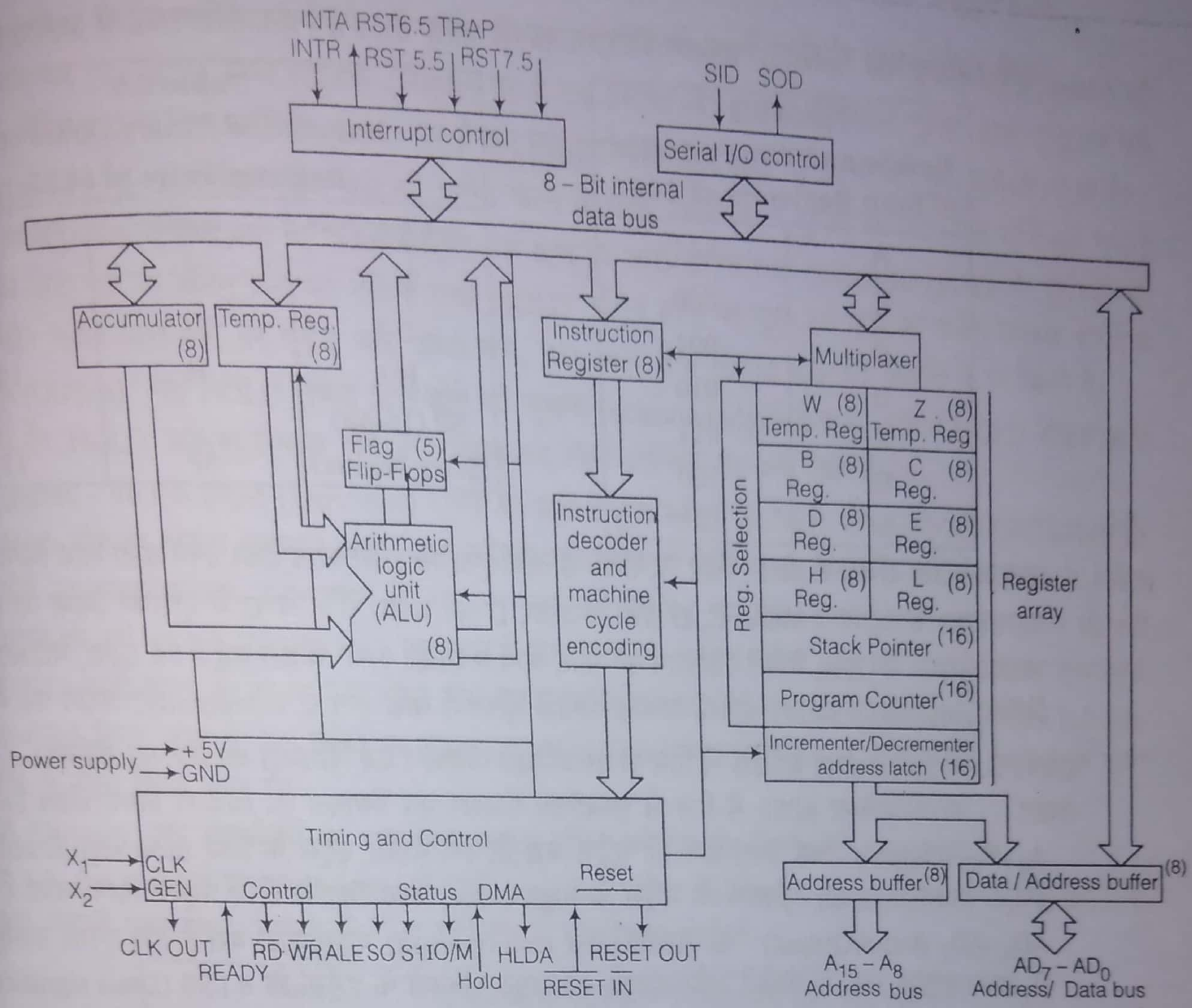
(iv) इन्सट्रक्शन डिकोडर (Instruction Decoder)

(v) जनरल परपज रजिस्टर्स (GPRs)

(vi) इन्टरप्ट कन्ट्रोल (Interrupt Control)

(vii) सीरियल I/O कन्ट्रोल (Serial Input/Output Control)

Scanned with CamScanner



चित्र **5.1** माइक्रोप्रोसेसर 8085 की संरचना

प्रश्न 2. अर्थमैटिक एवं लॉजिक यूनिट (ALU) के मुख्य भागों का वर्णन कीजिए।

उत्तर अर्थमैटिक एवं लॉजिक यूनिट एएलयू Arithmetic and Logic Unit (ALU) ALU माइक्रोप्रोसेसर में समस्त गणितीय एवं लॉजिक इन्सट्रक्शन्स एक्जीक्यूट करता है। ALU के निम्न मुख्य भाग हैं—

(a) एकुमुलेटर (Accumulator)

(b) टैम्पोरेरी रजिस्टर (Temporary Register)

(c) फ्लैग (Flags)

(a) **एकुमुलेटर** Accumulator एकुमुलेटर (ACC) एक 8-बिट रजिस्टर है। इसका प्रयोग विभिन्न गणितीय एवं लॉजिक क्रियाओं में किया जाता है। उदाहरण के लिए दो 8-बिट संख्याओं को जोड़ने की क्रिया में एक संख्या एकुमुलेटर में तथा दूसरी किसी मेमोरी अथवा रजिस्टर में हो सकती है। गणितीय क्रियाओं के परिणाम एकुमुलेटर में स्टोर होते हैं। यह रजिस्टर *A* कहलाता है तथा इसे ACC द्वारा भी प्रदर्शित करते हैं।

(b) **टैम्पोरेरी रजिस्टर** Temporary Register किसी गणितीय एवं लॉजिक क्रिया सम्पन्न होने के समय (during an arithmetic and logic operation) टैम्पोरेरी रजिस्टर का प्रयोग डाटा होल्ड (hold) करने के लिए किया जाता है।

(c) **फ्लैग** Flags माइक्रोप्रोसेसर 8085 में 5 फ्लैग होते हैं—

S, Z, AC, P तथा CY

फ्लैग, 5 फ्लिप-फ्लॉप का एक सैट होता है। इनका कार्य एकुमुलेटर तथा अन्य रजिस्टरों में डाटा का स्टेट्स प्रदर्शित करना हैं जिस समय माइक्रोप्रोसेसर में गणितीय एवं लॉजिक ऑपरेशन होते हैं उस समय टैम्पोरेरी रजिस्टर में डाटा तथा एकुमुलेटर में परिणाम स्टोर होता है। परिणाम के अनुसार फ्लैग सैट (Set) अथवा रिसैट (Reset)

Scanned with CamScanner

होते हैं। सैट होने का अर्थ '1' तथा रिसैट होने का अर्थ '0' है। इस प्रकार फ्लैग, एकुमुलेटर में स्टोर किये गये डाटा (परिणाम) की स्थिति प्रदर्शित करते हैं। चित्र 5.2 में एक फ्लैग रजिस्टर प्रदर्शित किया गया है। विभिन्न फ्लैग का विवरण निम्न प्रकार है—

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z	—	AC	—	P	—	CY

चित्र 5.2 फ्लैग प्वायन्टर्स

(i) **S-Sign Flag** यदि किसी गणितीय प्रचालन के पश्चात् प्राप्त परिणाम में आठवें बिट अर्थात् D_7 का मान 1 होता है तब Sign फ्लैग सैट (1) होता है। किसी बाइट (byte) में यदि D_7 का मान 1 है तब संख्या ऋणात्मक मानी जाती है तथा यदि शून्य है तब संख्या धनात्मक मानी जाती है। यदि गणितीय ऑपरेशन में संख्याओं के साथ कोई चिह्न (+ अथवा –) प्रयोग किया जाता है तब D_7 का कार्य केवल चिह्न प्रदर्शित करना है तथा शेष 7 बिट ($D_0 - D_6$) संख्या का परिमाण (magnitude) प्रदर्शित करते हैं।

(ii) **Z-Zero Flag** यदि ALU में किये गये ऑपरेशन का परिणाम शून्य है तब Zero फ्लैग सैट (1) होता है। यदि परिणाम शून्य नहीं है तब फ्लैग Reset (0) होता है। यह फ्लैग एकुमुलेटर तथा अन्य रजिस्टर में प्राप्त परिणाम के अनुसार बदलता रहता है।

(iii) **AC-Auxiliary Carry Flag** किसी गणितीय ऑपरेशन में जब D_3 अंक द्वारा कोई कैरी (Carry) उत्पन्न होती है तथा D_4 की ओर जाती है तब AC फ्लैग सैट होता है। यह फ्लैग केवल आन्तरिक BCD कोड के लिए प्रयोग होता है तथा किसी अन्य प्रचालन (जैसे jump operation) द्वारा इसके प्रोग्राम का क्रम परिवर्तित नहीं किया जा सकता है।

(iv) **P-Parity Flag** किसी गणितीय क्रिया के परिणाम में यदि 1_S की संख्या सम (even) होती है तब यह फ्लैग रिसैट हो जाता है।

(v) **CY-Carry Flag** यदि किसी गणितीय क्रिया के परिणाम में कैरी (Carry) प्राप्त होती है तब यह फ्लैग सैट (set) अन्यथा रिसैट (reset) होता है। घटाने की क्रिया में यह फ्लैग बॉरो फ्लैग (borrow flag) की भाँति कार्य करता है।

प्रश्न 3. इन्टेल माइक्रोप्रोसेसर 8085 का वर्णन कीजिए।

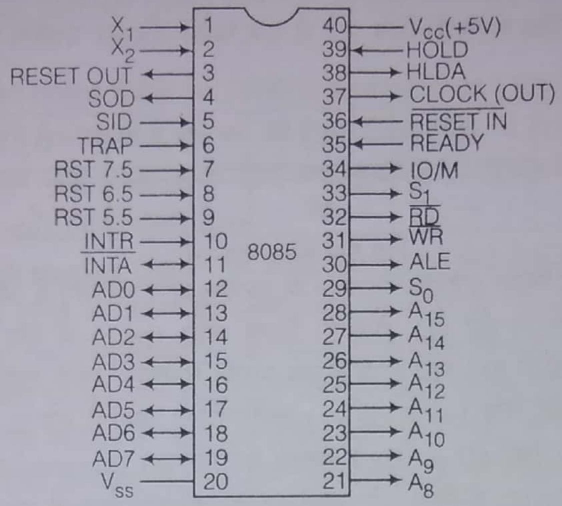
उत्तर इन्टेल माइक्रोप्रोसेसर 8085 का वर्णन निम्नलिखित है—

(i) 40 pin dual-in-line package (DIP)
(ii) NMOS device implemented with approximately 6200 transistors on a '164 × 222' mil chip. (1 mil = 0.001 inch)
(iii) Word size 8-bit
(iv) Requirement of supply – 5 volts d.c.
(v) Clock frequency = 3.125 (MHz) (8085A)
(vi) Clock cycle time = 320 ns (8085A)
(vii) Shortest instruction execution time = 1.3 microsecond (8085A)
(viii) Total basic instructions = 80
(ix) Total defined instructions or opcodes = 246
(x) Bus structure – 16 bit address bus, 8 bit data bus, [$AD_0 - AD_7$] multiplexes [$D_0 - D_7$ with $A_0 - A_7$]
(xi) Year of manufacturing – 1976
(xii) Manufacturer – intel corporation U.S.A.

Scanned with CamScanner

प्रश्न 4. Intel 8085 माइक्रोप्रोसेसर का पिन आरेख बनाइए।

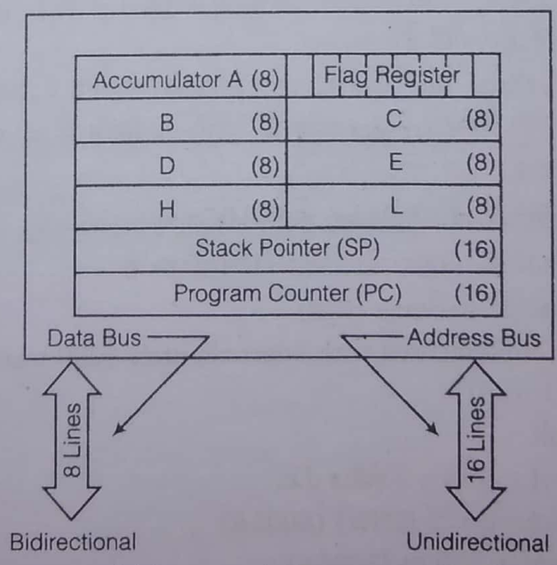
उत्तर



चित्र **5.3** Pin diagram of 8085 microprocessor

प्रश्न 5. माइक्रोप्रोसेसर 8085 के जनरल परपज रजिस्टर्स (General Purpose Register-GPRs) का वर्णन कीजिए।

उत्तर माइक्रोप्रोसेसर-8085 में 6 GPRs (General Purpose Registers) होते हैं जिन्हें B, C, D, E, H तथा L संकेतों द्वारा प्रदर्शित किया जाता है। यह सभी 8-बिट रजिस्टर्स हैं तथा माइक्रोप्रोसेसर में छोटे आकार की मेमोरी की भाँति कार्य करते हैं। प्रोग्राम एक्जीक्यूशन के समय यह रजिस्टर 8-बिट डाटा स्टोर करते हैं। चित्र 5.4 में 8085 के GPRs प्रदर्शित किये गये हैं।



चित्र **5.4** माइक्रोप्रोसेसर 8085 के जनरल परपज रजिस्टर्स (GPRs)

8-बिट डाटा ऑपरेशन के लिए यह रजिस्टर्स अलग-अलग प्रयोग किये जाते हैं तथा 16-बिट डाटा ऑपरेशन के लिए एक साथ दो-दो रजिस्टर, रजिस्टर पेयर (Register pair) की भाँति प्रयुक्त किये जाते हैं। रजिस्टर पेयर (pair) की भाँति प्रयोग हेतु केवल निम्न पेयर्स ही अनुमन्य (allowed) हैं—

Register pair	B-C
Register pair	D-E
Register pair	H-L

Scanned with CamScanner

सभी GPRs तथा रजिस्टर पेयर्स के निश्चित बानरी कोड हैं। इन्हें निम्न तालिकाओं में प्रदर्शित किया गया है—

Table 5.1
Names and Binary Codes of 8-Bit Registers of 8085

A	111
B	000
C	001
D	010
E	011
H	100
L	101

Table 5.2
Names and Binary Codes of Register Pairs of 8085

Register Pair	Name	Binary Code
B-C	B	00
D-E	D	01
H-L	H	10
SP (16-Bit) (Stack Pointer)	SP	11

जनरल परपज रजिस्टर्स प्रोग्राम किये जा सकते हैं अर्थात् उचित इन्सट्रक्शन देकर इनमें डाटा लोड किया जा सकता उदाहरणत: इन्सट्रक्शन MOV, C, D द्वारा रजिस्टर D का डाटा, रजिस्टर C में ट्रांसफर किया जा सकता है। माइक्रोप्रोसेसर में GPR, मेमोरी लोकेशन की भाँति कार्य करते हैं, अन्तर केवल यह है कि GPR, माइक्रोप्रोसेसर में निर्मित (internally built) होते हैं जबकि मेमोरी युक्तियाँ बाह्य रूप से (externally) प्रयोग की जाती है।

प्रश्न 6. माइक्रोप्रोसेसर 8085 में पिन से सम्बन्धित फंक्शन एवं सिगनल का विवरण दीजिए।

उत्तर माइक्रोप्रोसेसर 8085 में पिन से सम्बन्धित फंक्शन एवं सिगनल का विवरण निम्नलिखित है—

A_8–A_{15} (Output) यह उच्च क्रम की एड्रैस बस है। इसे मेमोरी एड्रैस के लिए most significant bits अ इनपुट/आउटपुट (I/O) युक्तियों के एड्रैस के high order, आठ बिट के लिए प्रयुक्त किया जाता है।

AD_0–AD_7 (Input/Output) यह मल्टीप्लैक्स्ड (multiplexed) एड्रैस/डाटा बस है तथा इनका उपयोग मशी के प्रथम क्लॉक चक्र में 'मेमोरी एड्रैस/इनपुट-आउटपुट युक्तियों के एड्रैस' के 8 LSB (Least significant bi लिए किया जाता है। द्वितीय एवं तृतीय क्लॉक चक्र में इनका उपयोग डाटा ट्रांसफर के लिए किया जाता है।

ALE [Address Latch Enable (Output)] यह एक positive going सिगनल है तथा यह प्रत्येक ऑपरेशन होने के समय (मशीन Cycle के प्रारम्भ में) उत्पन्न होता है। ALE सिगनल प्रदर्शित करता है कि डाटा बस (A AD_0) पर एड्रैस के low order bits हैं अथवा नहीं। इस सिगनल का प्रयोग मल्टीप्लैक्स्ड A/D बस से low एड्रैस अलग (Latch) करने तथा A_7–A_0 एड्रैस लाइन का एक अलग सैट उत्पन्न (generate) करने के लि जाता है।

S_0, S_1 (Output) यह माइक्रोप्रोसेसर द्वारा प्रेषित स्टेट्स सिगनल हैं जो विभिन्न ऑपरेशन में विभेद (differen करते हैं। S_1 तथा S_0 का High (1) अथवा Low (0) स्टेट्स निम्न ऑपरेशन की सूचना देता है—

S_1	S_0	Operation
0	0	HALT
0	1	WRITE
1	0	READ
1	1	FETCH

$\overline{RD}$ Read (Output) यह Read कन्ट्रोल सिगनल है तथा Read ऑपरेशन के लिए यह Active Low होता है। सिगनल प्रदर्शित करता है कि सलैक्ट की गयी मेमोरी अथवा I/O युक्ति को Read किया जाना है तथा डाटा बस डाटा उपलब्ध है।

$\overline{WR}$ Write (Output) यह Write कन्ट्रोल सिगनल (Active Low) है। यह सिगनल प्रदर्शित करता है कि डाटा पर उपलब्ध डाटा किसी सलैक्ट की गयी मेमोरी अथवा I/O युक्ति में Write किया जाना है।

Scanned with CamScanner

READY (Input) यह संकेत किसी पैरीफैरल युक्ति की डाटा ट्रांसफर करने की स्थिति की सूचना माइक्रोप्रोसेसर को देता है। यदि यह (READY) उच्च लॉजिक (High) पर है तब इसका अर्थ है कि पैरीफैरल युक्ति डाटा ट्रांसफर कर सकती है। यदि READY निम्न लॉजिक (Low) पर है तब माइक्रोप्रोसेसर READY के उच्च होने तक प्रतीक्षा करता है।

HOLD (Input) यह डाटा तथा एड्रैस बस को प्रयोग करने के लिए किसी अन्य युक्ति के प्रतीक्षारत रहने की सूचना देता है। 'HOLD' प्रार्थना प्राप्त होने पर माइक्रोप्रोसेसर उस क्षण पर जारी मशीन चक्र (machine cycle) के पूर्ण होते ही एड्रैस एवं डाटा बस का प्रयोग बन्द कर देता है तथा 'HOLD' संकेत हटने पर पुन: इस बस का प्रयोग प्रारम्भ कर देता है। HOLD संकेत प्राप्त होने पर एड्रैस बस, डाटा बस, RD, WR तथा IO/M ट्राइस्टेट स्थिति में आ जाते हैं।

HALDA (Output) यह HOLD संकेत प्राप्त होने की सूचना (acknowledge) देता है, अर्थात् HOLD संकेत प्राप्त हो चुका है। 'HOLD' प्रक्रिया समाप्त होने के पश्चात् यह निम्न लॉजिक पर आ जाता है।

INTR (Input) यह एक इन्ट्रप्ट (interrupt) संकेत है। सभी इन्ट्रप्ट संकेतों में INTR संकेत की प्रायिकता निम्नतम है। इसके High होने पर प्रोग्राम काउन्टर (PC) द्वारा की जा रही गणना में वृद्धि (increment) नहीं होती। INTR संकेत प्राप्त होने पर माइक्रोप्रोसेसर, इन्सट्रक्शन के सामान्य कार्य को स्थगित कर देता है तथा वर्तमान में प्रोसेस हो रही इन्सट्रक्शन को पूर्ण करने के पश्चात् CALL इन्ट्रक्शन में चला जाता है। INTR संकेत प्राप्त कर माइक्रोप्रोसेसर एक अन्य संकेत INTA जारी करता है। प्रयुक्त सॉफ्टवेयर द्वारा आवश्यकतानुसार INTR, संकेत इनेबल अथवा डिसेबल (enable or disable) किया जाता है। इस संकेत का प्रयोग I/O युक्तियों द्वारा माइक्रोप्रोसेसर का समय नष्ट किये बिना डाटा ट्रांसफर के लिए किया जाता है।

$\overline{\text{INTA}}$ (Output) यह माइक्रोप्रोसेसर द्वारा प्राप्त INTR संकेत स्वीकार कर लिये जाने का सूचक है।

RST 5.5, 6.5, 7.5 एवं TRAP (Input) ये सभी interrupt संकेत हैं। किसी interrupt संकेत को पहचानने के पश्चात् अगली इन्सट्रक्शन, मेमोरी की निम्न लोकेशन से एक्जीक्यूट होती है—

Line	Next इन्सट्रक्शन एक्जीक्यूट होने का स्थान
TRAP	0024
RST 5.5	002C
RST 6.5	0034
RST 7.5	003C

उपरोक्त में से RST 5.5, 6.5 तथा 7.5 पुन: प्रारम्भ करने हेतु (restart) interrupts हैं। इनके द्वारा माइक्रोप्रोसेसर के अन्दर restart की प्रक्रिया स्वत: प्रारम्भ हो जाती है। इनमें से प्रत्येक को प्रोग्राम किया जा सकता है। सभी interrupts में TRAP की प्रायिकता उच्चतम है। इस interrupt को मास्क नहीं किया जा सकता तथा यह किसी अन्य मास्क अथवा interrupt enable से प्रभावित नहीं होता। Interrupts की प्रायिकता क्रमिक रूप से निम्न है—

Interrupt	Priority
TRAP	1
RST 7.5	2
RST 6.5	3
RST 5.5	4

$\overline{\text{RESET IN}}$ (Input) यह संकेत प्रोग्राम काउन्टर को शून्य पर लाता है। यह Interrupt इनेबल तथा HLDA फ्लिप-फ्लॉप को भी रिसैट करता है। यह इन्सट्रक्शन रजिस्टर (IR) के अतिरिक्त अन्य किसी फ्लैग को प्रभावित नहीं करता। RESET प्रयुक्त किये जाने की अवधि तक माइक्रोप्रोसेसर RESET स्थिति में ही रहता है।

RESET OUT (Output) यह संकेत माइक्रोप्रोसेसर को RESET किये जाने की सूचना देता है।

Scanned with CamScanner

X_1, X_2 (Input) ये टर्मिनल किसी बाह्य दोलित्र (oscillator) से संयोजित किये जाते हैं। बाह्य दोलित्र माइक्रोप्रोसेसर के आन्तरिक परिपथ को प्रचालित कर माइक्रोप्रोसेसर के ऑपरेशन के लिए एक उचित क्लॉक (clock) की व्यवस्था करता है।

CLK (Input) इस टर्मिनल पर क्लॉक सिगनल उपलब्ध होता है। इसका उपयोग अन्य डिजिटल ICs के प्रचालन में किया जा सकता है। इस क्लॉक सिगनल की आवृत्ति माइक्रोप्रोसेसर की प्रचालन आवृत्ति के बराबर होती है।

SID (Input) यह सीरियल इनपुट के लिए डाटा लाइन है।

SOD (Output) यह सीरियल आउटपुट के लिए डाटा इन है।

V_{CC} + 5V सप्लाई।

V_{SS} Ground reference line

प्रश्न 7. माइक्रोप्रोसेसर के टाइमिंग एवं कन्ट्रोल सिगनल को समझाइए।

उत्तर 8085 **के टाइमिंग एवं कन्ट्रोल सिगनल** Timing and Control Signals for 8085 टाइमिंग एवं कन्ट्रोल यूनिट माइक्रोप्रोसेसर को दिये गये इन्ट्रक्शन के एक्जीक्यूशन (execution) एवं पैरीफैरल युक्तियों के नियन्त्रण के लिए टाइमिंग सिगनल (timing signal) उत्पन्न करता है। किसी इन्सट्रक्शन को एक्जीक्यूट करने के लिए माइक्रोप्रोसेसर सर्वप्रथम मेमोरी से इन्सट्रक्शन प्राप्त (fetch) करता है। तत्पश्चात् उसको एक्जीक्यूट करता है। किसी इन्सट्रक्शन को एक्जीक्यूट करने में लिया गया समय इन्सट्रक्शन चक्र (Instruction Cycle) कहलाता है। इन्सट्रक्शन चक्र में दो चक्र सम्मिलित होते हैं—

(i) फैच चक्र (Fetch Cycle)

(ii) एक्जीक्यूट चक्र (Execute Cycle)

फैच चक्र (FC) माइक्रोप्रोसेसर द्वारा फैच ऑपरेशन में लिया गया समय है। इस समय में मेमोरी से इन्सट्रक्शन का मशीन कोड (opcode) प्राप्त किया जाता है। यह समय निश्चित होता है।

एक्जीक्यूट चक्र (EC) में लगा समय इन्सट्रक्शन के आकार पर निर्भर करता है। इस प्रकार किसी इन्सट्रक्शन के एक्जीक्यूशन में व्यतीत समय (IC) को निम्न प्रकार प्रदर्शित किया जा सकता है—

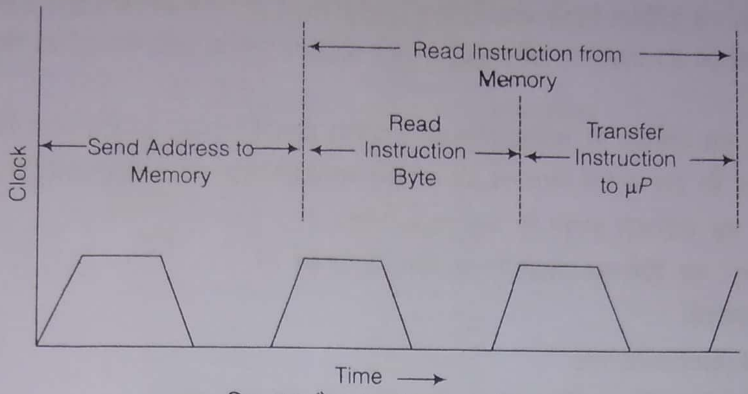
Instruction Cycle (IC) = Fetch Cycle (FC) + Execute Cycle (EF)

प्रश्न 8. फैच ऑपरेशन (Fetch Operation) का वर्णन कीजिए।

उत्तर **फैच ऑपरेशन** Fetch Operation फैच ऑपरेशन में माइक्रोप्रोसेसर, मेमोरी से इन्सट्रक्शन का प्रथम बाइट (first byte of instruction) प्राप्त करता है। यह बाइट, ऑपरेशन कोड (opcode) होता है।

उदाहरणत: इन्सट्रक्शन MVI B,05 दो बाइट इन्सट्रक्शन हैं। इसका प्रथम बाइट MVI B है। फैच ऑपरेशन में माइक्रोप्रोसेसर इस बाइट का ऑपरेशन कोड, जो कि '06' है, मेमोरी से फैच (Fetch) करता है। हम जानते हैं कि माइक्रोप्रोसेसर में प्रोग्राम काउन्टर (PC) अगली (next) एक्जीक्यूट की जाने वाली इन्सट्रक्शन के एड्रैस का ब्यौरा (track) रखता है अत: फेच चक्र प्रारम्भ होने के क्षण पर प्रोग्राम काउन्टर के Content मेमोरी में लोड हो जाते हैं। मेमोरी इस एड्रैस को पढ़ती हा। तथा माइक्रोप्रोसेसर को ट्रांसफर करती है। चित्र 5.4 (a) में फैच ऑपरेशन दिखायश गया है। ध्यान दीजिए कि एड्रैस ट्रांसफर होने की क्रिया एक क्लॉक चक्र (clock cycle) में सम्पन्न होती है। अगले (next) दो क्लॉक चक्रों में दो ऑपरेशन होते हैं। पहले चक्र में मेमोरी उस लोकेशन से इन्सट्रक्शन पढ़ती है जिसका एड्रैस उसने पिछले (previous) क्लॉक चक्र में प्राप्त किया है। दूसरे चक्र में इन्सट्रक्शन, मेमोरी से माइक्रोप्रोसेसर को ट्रांसफर होती है। यह पूरी क्रिया फैच ऑपरेशन कहलाती है तथा इसमें लगा समय फैच चक्र (fetch cycle) कहलाता है। ऑप-कोड फैच करने की सम्पूर्ण प्रक्रिया में 3 क्लॉक चक्रों (three clock cycles) के तुल्य समय व्यतीत होता है।

Scanned with CamScanner



चित्र **5.5** फैच चक्र (Fetch Cycle)

इस प्रकार माइक्रोप्रोसेसर से एड्रैस प्राप्त करने के पश्चात् मेमोरी, उपरोक्त वर्णन के अनुसार, इस क्रिया में दो क्लॉक चक्र लेती है। यदि मेमोरी की गति कम (slow) है तब इसे और अधिक समय भी लग सकता है। इस स्थिति में माइक्रोप्रोसेसर को तब तक प्रतीक्षा करनी पड़ती है जब तक कि मेमोरी से ऑपरेशन कोड (opcode) प्राप्त नहीं हो जाता है। माइक्रोप्रोसेसर द्वारा ऑपरेशन कोड के प्राप्त होने की प्रतीक्षा में व्यतीत किया गया समय Wait Cycle कहलाता है। अधिकतर माइक्रोप्रोसेसर में धीमी गति वाली मेमोरी से कम्पयूनिकेट करने के लिए wait cycles की व्यवस्था होती है।

प्रश्न 9. माइक्रोप्रोसेसर 8085 और माइक्रोप्रोसेसर 8086 के बीच में अन्तर बताइए।

माइक्रोप्रोसेसर 8085 और 8086 में अन्तर

	Comparison in terms of	**Intel 8085**	**8086**
1.	Manufacturer	Intel Corporation	Intel Corporation
2.	Manufacturing Year	1976	1978
3.	No. of Pins	40	40
4.	Word Size	8	16
5.	Clock Frequency	3.125 MHz	5 MHz
6.	Clock Time	320 nenoseconds	0.2 microseconds
7.	Shortest Instruction Execution Time	1.3 microseconds	0.4 microseconds
8.	Bus Structure	16-बिट एड्रेस बस, 8-बिट डाटा बस ($AD_0 - AD_7$) एड्रेस/डाटा Multiplexing	20-बिट एड्रेस बस 16-बिट डाटा बस 16-बिट डाटा बस By multiplexing
	INTERRUPTS		
9.	General Purpose	INTR Forced Instruction	INTR
10.	Non-maskable	TRAP	NMI
11.	Maskable	RST 7.5, RST 6.5 तथा RST 5.5	INTR is masked if IF = Reset (IF = Interrupt Flag)
12.	Software	RSTO ... RST7	INT or INTO

Scanned with CamScanner

	Architecture		
13.	Internal registers	8-बिट रजिस्टर्स— ACC, B, C, D, E, H, L तथा SFR; 16-बिट रजिस्टर्स	16-बिट रजिस्टर्स: स्टैक प्वाइन्टर, बेस प्वाइन्टर, सोर्स इन्डेक्स रजिस्टर, डेस्टीनेशन इन्डेक्स रजि०, कोड सेगमेन्ट रजि०, रजि०, एकस्ट्रा सेगमेन्ट रजि०, इन्स्ट्रक्शन प्वाइन्टर, स्टेटस फ्लैग रजिस्टर, AX, BX, CX तथा DX जनरल परपज रजिस्टर
	Control Signal		
14.	Wait State Signal	READY	READY
15.	Read/Write Signal	$\overline{RD}$ & $\overline{WR}$ (Two lines)	$\overline{RD}$ & $\overline{WR}$ (Two lines)
16.	Status Signals	S_1, S_0, &$IO/\overline{M}$	$\overline{DEN}$, $DT/\overline{R}$, $M/\overline{IO}$ $\overline{DEN}$ ($\bar{S}_0$), $DT/\overline{R}$ (S_1)
17.	Reset	$\overline{RESET\ IN}$ & RESET OUT	RESET
18.	Serial I/O	SID & SOD	None
19.	DMA Control lines	HOLD & HLDA	HOLD & HLDA
20.	Dynamic Memory Control	None	None
21.	I/O-mapped I/O Status Flags	8-बिट I/O एड्रेस	Yes
22.	Different Flags Used	Sign (S), Zero (Z), Parity (P), Carry (C), Auxilary Carry (AC)	C, P, AC, Z, S, T, I, D, O. D : Direction Flag T : Trap Flag I : Interrupt Flag O : Over Flow Flag

Scanned with CamScanner

6 मल्टी प्रोसेसर प्रणाली का निर्माण
Architecture of Multi Processor Systems

खण्ड 'अ' : अतिलघु उत्तरीय प्रश्न

प्रश्न 1. पैरेलल कम्प्यूटिंग की परिभाषा लिखिए।

उत्तर पैरेलल कम्प्यूटिंग एक ऐसी कम्प्यूटिंग है जिसमें दो या दो से अधिक प्रोसेसरों का प्रयोग एक अकेली समस्या को हल करने में किया जाता है।

प्रश्न 2. पैरेलल कम्प्यूटिंग कितने प्रकार की होती है?

उत्तर पैरेलल कम्प्यूटिंग दो प्रकार की होती है—

(i) सामान्य उद्देश्य के लिए तथा (ii) विशेष उद्देश्य के लिए

प्रश्न 3. पाइपलाइनिंग का कार्य बताइए।

उत्तर पाइपलाइनिंग सिस्टम की कार्यक्षमता को बढ़ाती है।

प्रश्न 4. पाइपलाइन में निर्देश (Instruction) को एक्जीक्यूट करने के लिए कितनी स्टेज होती है?

उत्तर 5.

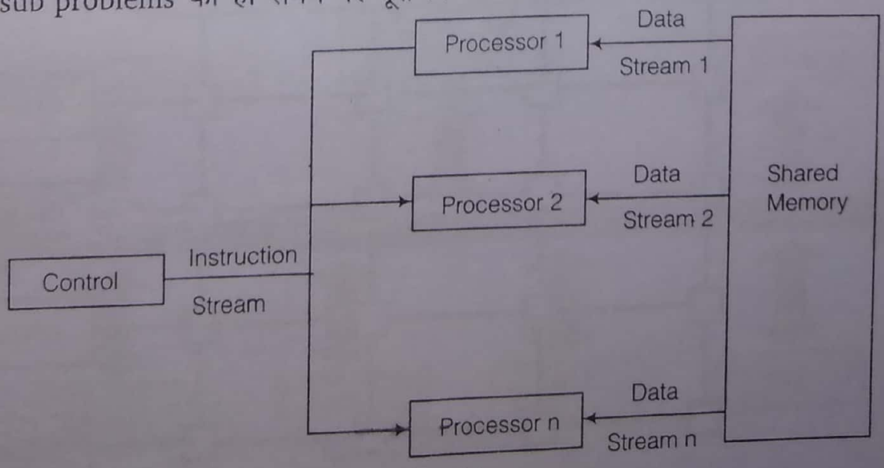
प्रश्न 5. पाइपलाइनिंग का एक लाभ लिखिए।

उत्तर यह प्रक्रिया में लगने वाले समय को कम करता है।

खण्ड 'ब : लघु एवं दीर्घ उत्तरीय प्रश्न

प्रश्न 1. पैरेलल प्रक्रिया (Parallel Processing) क्या है? उदाहरण सहित बताइए। इसके लाभ लिखिए।

उत्तर "Parallel Computing एक ऐसी कम्प्यूटिंग है जिसमें दो या दो से अधिक प्रोसेसरों का प्रयोग एक सिंगल समस्या को हल करने में किया जाता है"। Parallel Computing में जो कार्य होता है उसे प्रोसेसरों के मध्य विभाजित कर लिया जाता है अर्थात् जो मुख्य Operation (Problem) होती हैं उसे Sub Problems में विभाजित कर लिया जाता है और प्रत्येक sub problems को ही समय पर पूरा किया जाता है।



चित्र 6.1

Scanned with CamScanner

पहले के समय में Serial Computing का प्रयोग किया जाता था। परन्तु इसमें बहुत समय लगता था क्योंकि इस Computing में एक समय में केवल एक Operation ही प्रोसेस हो पाता था। जिसके कारण दूसरे ऑपरेशन को परफॉर्म होने के लिए इन्तजार करना पड़ता था।

उदाहरणार्थ माना कि इंटरव्यू देने कम्पनी में 100 अभ्यर्थी आये हैं और इंटरव्यू लेने के लिए केवल एक ही Interviewer है। वह एक समय में केवल एक व्यक्ति का इंटरव्यू ले रहा है तथा और व्यक्ति इन्तजार कर रहें हैं। इस प्रकार 100वें अभ्यर्थी का इंटरव्यू होने में शाम हो जायेगी। परन्तु अगर 4 Interviewer आयें हैं और चारों Interviewer अलग-अलग अभ्यर्थियों का इंटरव्यू ले रहे हैं तो 100वें अभ्यर्थी का इंटरव्यू भी दिन में ही खत्म हो जायेगा जिससे समय की बचत होगी अर्थात् Parallel Computing का मुख्य उद्देश्य कार्य को जल्दी समाप्त करके समय की बचत करना है। Parallel Computing के लाभ निम्नलिखित हैं—

1. इससे पैसे और समय की बचत होती है।
2. बहुत सारी समस्याएँ तो बहुत जटिल तथा बड़ी होती हैं तथा इनको केवल एक सिंगल प्रोसेसर से हल कर पाना मुश्किल होता है, इसलिए Parallel Computing का प्रयोग किया जाता है।
3. Serial Computing में केवल एक प्रोसेसर होता है जो कि एक समय में केवल एक ही कार्य करता है जबकि Parallel Computing द्वारा एक समय में बहुत सारे कार्य हो सकते हैं।
4. इसके द्वारा हम वाइड एरिया नेटवर्क या इन्टरनेट का प्रयोग भी कर सकते हैं।
5. आजकल जो PC तथा लैपटॉप होते हैं Parallel Computing पर आधारित होते हैं जिनमें बहुत सारे Processors होते हैं, जिन्हें हम Core (कोर) भी कहते हैं।

प्रश्न 2. पैरेलल कम्प्यूटिंग आर्किटेक्चर के प्रकारों का वर्णन कीजिए।

उत्तर पैरेलल कम्प्यूटिंग आर्किटेक्चर निम्न दो प्रकार का होता है—

1. **सामान्य उद्देश्य के लिए** For General Purpose सामान्य उद्देश्य आर्किटेक्चर वे होते हैं जिनका प्रयोग हम आजकल किसी सामान्य कार्य में करते हैं। इन आर्किटेक्चर के अन्तर्गत निम्न तीन आर्किटेक्चर आते हैं—
   (i) सिंक्रोनस (Synchronous), (ii) डाटा प्रवाह (Data Flow) तथा (iii) पाइप लाइन (Pipeline)।
2. **विशिष्ट उद्देश्य के लिए** For Special Purpose विशिष्ट उद्देश्य आर्किटेक्चर का प्रयोग किसी विशेष उद्देश्य के लिए बड़े-बड़े कम्प्यूटरों में किया जाता है। इसके अन्तर्गत निम्न आर्किटेक्चर आते हैं—
   (i) असिंक्रोनस (asynchronous), (ii) सिस्टोलिक (Systolic)।

प्रश्न 3. पाइपलाइनिंग (Pipelining) से आप क्या समझते हैं? इसकी अवस्थाएँ (Stages) एवं लाभ लिखिए।

उत्तर **पाइपलाइनिंग** Pipelining पाइपलाइनिंग एक ऐसी प्रक्रिया है जिसमें एक Operation/Problem को Sub Problems में विभाजित कर लिया जाता है और सभी sub problems अलग-अलग सेगमेंट में execute होती है तथा सारे सेगमेंट्स एक साथ execute होते हैं।

दूसरे शब्दों में कहें तो, "Pipelining एक ऐसी प्रक्रिया है जिसमें सीपीयू के हार्डवेयर को इस प्रकार व्यवस्थित किया जाता है जिससे कि सीपीयू की पूरी Execution तथा Performance बढ़ जाती है।"

उदाहरणार्थ माना कि एक बाइक का एक प्लांट है, उसमें तीन Stages हैं जिनसे बाइक को गुजरना पड़ता है। ये तीन स्टेज हैं—बाइक की बॉडी बनाना, इंजन बनाना तथा बाइक को पेंट करना। और प्रत्येक स्टेज जो है वह अपना कार्यखत्म करने में दो घंटे का वक्त लेती है।

अब बिना Pipelining ऑपरेशन (Sequential ऑपरेशन) में, पहले स्टेज में प्लांट में बाइक की बॉडी बनायी जाती है और 2 घंटे बाद यह दूसरे स्टेज में चली जाती है जहाँ इसका इंजन बनता है अब स्टेज 1 में कोई कार्य नहीं हो रहा है। जब बाइक तीसरे स्टेज में चली जाती है तो स्टेज 1 तथा स्टेज 2 में कोई कार्य नहीं होता है। परन्तु Pipelining में, जब बाइक स्टेज 2 में जाती है तो स्टेज 1 में दूसरी बाइक की बॉडी बनाई जाती है। इसी प्रकार जब बाइक स्टेज 3 में होती है तो स्टेज 1 तथा स्टेज 2 में दूसरी बाइक की बॉडी तथा इंजन का कार्य हो रहा होता है।

Scanned with CamScanner

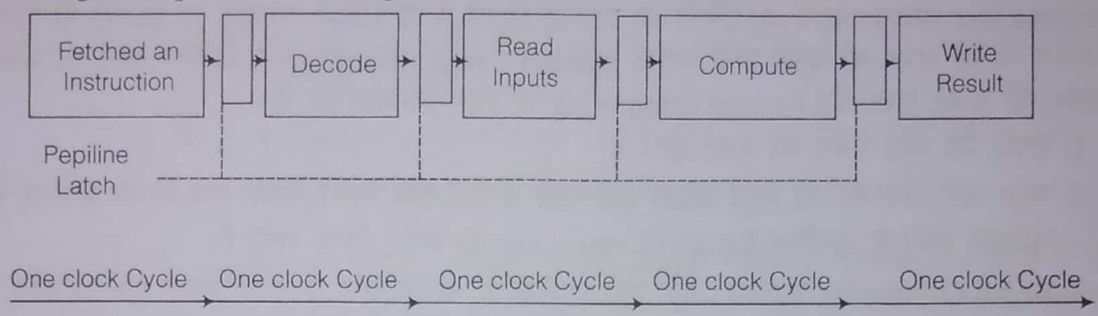
इसलिए हमें प्रत्येक 2 घंटे बाद स्टेज 3 में बाइक पेंटिंग के लिए आ जाती है और कोई स्टेज खाली नहीं होती, प्रत्येक स्टेज में कार्य हो रहा होता है तो हमने एक बाइक को बनाने में समय लगाया =

बिना Pipelining के (Sequential में) = 2 घंटे + 2 घंटे + 2 घंटे = 6 घंटे

Pipelining में = 2 घंटे

इससे यह पता चलता है कि Pipelining सिस्टम की कार्यक्षमता को बढ़ा देता है।

Pipeline Stages : Pipelined five Stages Processor



चित्र **6.2** Pipeline Stages

Pipeline में Instruction को Execute करने के लिए 5 stages होती हैं। जोकि निम्न हैं—

1. **Instruction Fetch** प्रथम स्टेज में, सीपीयू जो है वह मेमोरी में एड्रैस से इंस्ट्रक्शन को read करता है।
2. **Instruction Decode** दूसरी स्टेज में Instruction डिकोड होता है तथा Instruction में प्रयोग किये गये रजिस्टरों में से वैल्यू को प्राप्त किया जाता है।
3. **Execution** तीसरी स्टेज में सभी ALU ऑपरेशनों को पूरा किया जाता है। जो भी गणनाएँ होती हैं वह ALU के द्वारा की जाती हैं।
4. **Memory/IO** चौथी स्टेज में, इंस्ट्रक्शन में प्रस्तुत मेमोरी में/से मेमोरी Operands को Read तथा Write किया जाता है तथा यह प्रोसेसर से इनपुट तथा आउटपुट के लिए भी जिम्मेदार होता है।
5. **Write Back** कम्प्यूटर द्वारा Fetch की गयी वैल्यू को मेमोरी में उपस्थित रजिस्टर को वापस Write की जाती है।

पाइपलाइनिंग के लाभ Advantages of Pipelining Pipelining के लाभ निम्नलिखित हैं—

1. यह Processing में लगने वाले समय को कम कर देता है।
2. यह सिस्टम के Throughput (प्रवाहक्षमता) को बढ़ा देता है।
3. इससे सिस्टम की विश्वसनीयता बढ़ती है।

प्रश्न 4. मल्टीप्रोसेसर के सामान्य लक्षणों का वर्णन कीजिए।

उत्तर **मल्टीप्रोसेसर के सामान्य लक्षण** Basic Characteristics of Multiprocessor मल्टीप्रोसेसर के सामान्य लक्षण निम्नलिखित हैं—

(i) एक मल्टीप्रोसेसर का दो या दो से अधिक CPU मेमोरी और इनपुट-आउटपुट उपकरण के साथ संम्बन्ध होता है।

(ii) मल्टीप्रोसेसर सिस्टम में एक टर्म प्रोसेसर होता है। यह सेंट्रल प्रोसेसिंग यूनिट (CPU) या इनपुट/आउटपुट प्रोसेसर (IOP) होता है।

(iii) मल्टीप्रोसेसर को बहु-निर्देशक धारा (Multiple Instruction Stream) या बहुत-डाटा धारा (MIMD) प्रणाली में वर्गीकृत किया जाता है।

(iv) एक मल्टीप्रोसेसर सिस्टम एक ऑपरेटिंग सिस्टम द्वारा नियंत्रित किया जाता है जो प्रोसेसर के बीच परस्पर क्रिया प्रदान करता है और सिस्टम के सभी घटक समस्या के समाधान में इसका सहयोग करते हैं।

(v) मल्टीप्रोसेसर मल्टीप्रोसेसिंग सिस्टम की विश्वसनीयता में सुधार करता है जिससे सिस्टम के किसी भाग के फेल होने पर सिस्टम पर कोई प्रभाव न पड़े।

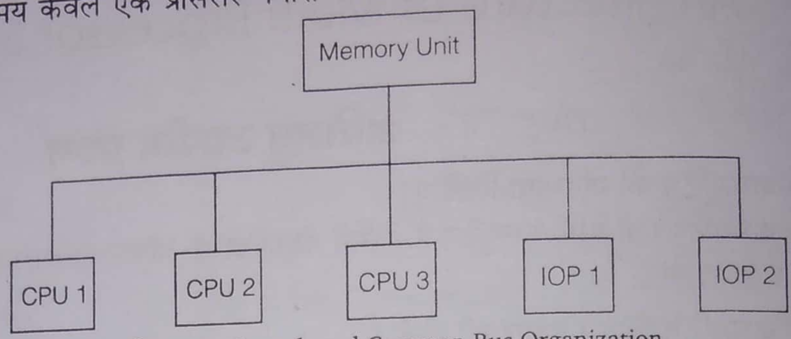
(vi) यदि कोई गलती एक प्रोसेसर के विफल होने का कारण बनती है तो विफेल प्रोसेसर के कार्यों को अन्य प्रोसेसर की सहायता से कराया जाता है।

Scanned with CamScanner

(vii) मल्टीप्रोसेसर संगठन प्रणाली से अनेक लाभ प्राप्त होते हैं।

प्रश्न 5. टाइम-शेयर्ड कॉमन बस (Time Shared Common Bus) सिस्टम का वर्णन कीजिए।

उत्तर टाइम-शेयर्ड कॉमन बस Time Shared Common Bus एक कॉमन बस प्रणाली में एक मेमोरी पाथ पर एक कॉमन पथ के माध्यम से अनेक प्रोसेसर होते हैं। नीचे चित्र में पाँच प्रोसेसरों को एक मेमोरी यूनिट से जुड़े हुए दिखाया गया है। किसी भी समय केवल एक प्रोसेसर मेमोरी या किसी अन्य प्रोसेसर के साथ संचार कर सकता है।

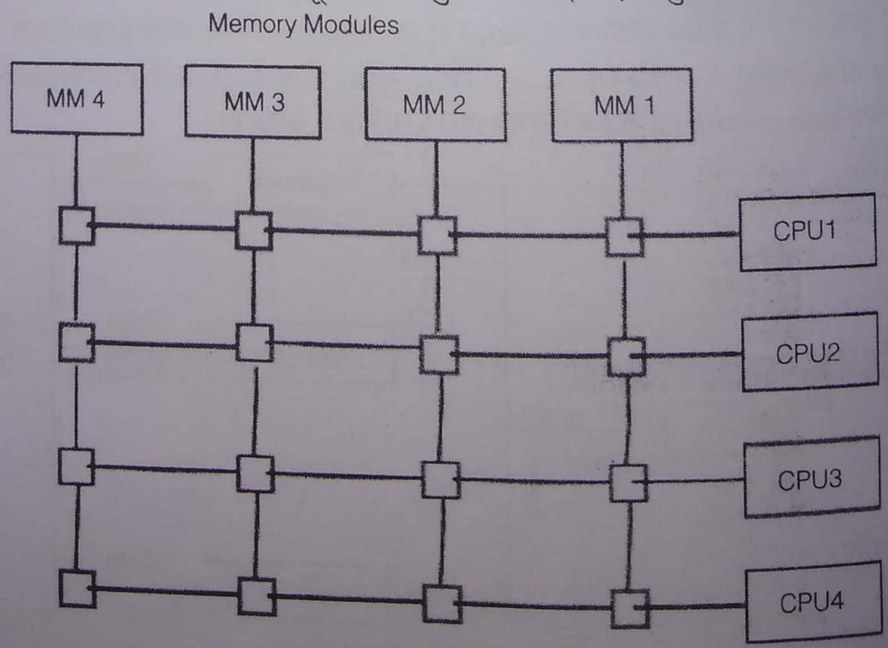


चित्र 6.3 Time-shared Common Bus Organization

ट्रांसफर ऑपरेशन उस समय बस के नियंत्रण में रहने वाले प्रोसेसर द्वारा किया जाता है। ट्रांसफर आरम्भ करने से पहले प्रोसेसर को बस की उपलब्धता की स्थिति निर्धारित करनी चाहिए और बस उपलब्ध होने के बाद ही प्रोसेसर गंतव्य इकाई को ट्रांसफर आरम्भ करने के लिए सम्बोधित करता है। डाटा किस इकाई को ट्रांसफर होना है उसके लिए यह एक कमाण्ड जारी करता है। प्राप्त इकाई बस में अपने पते को पहचानती है और प्रेषक से नियंत्रण संकेतों को सूचित करती है जिसके बाद ट्रांसफर की प्रक्रिया शुरू हो जाती है।

प्रश्न 6. क्रॉसबार (Crossbar Switch) का संक्षेप में वर्णन कीजिए।

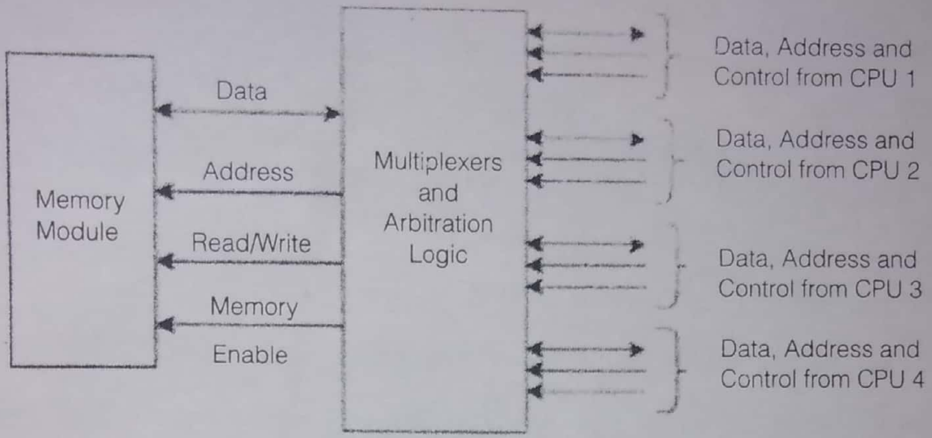
उत्तर क्रासबार स्विच Crossbar Switch क्रॉसबार स्विच सिस्टम में क्रॉसप्वाइंट होते हैं जो मेमोरी मॉडयूल और प्रोसेसर बसों के रास्तों के बीच चौराहों पर रखे जाते हैं। प्रत्येक क्रॉसपॉइंट में, छोटा वर्ग एक स्विच का प्रतिनिधित्व करता है जो प्रोसेसर से मेमोरी मॉडयूल तक का मार्ग प्राप्त करता है। प्रत्येक स्विच प्वाइंट में मेमोरी और प्रोसेसर के बीच ट्रांसफर पथ सेट करने के लिए नियंत्रण तर्क होता है। यह उस पते की गणना करता है जिसे प्राप्त करने के लिए बस में रखा गया है कि उसके विशिष्ट मॉडयूल को सम्बोधित किया जा रहा है या नहीं। इसके अलावा, यह पूर्व निर्धारित प्राथमिकता के आधार पर एक ही मेमोरी मॉडयूल तक पहुँच के लिए कई अनुरोधों को समाप्त करता है।



चित्र 6.4 Crossbar Switch System

Scanned with CamScanner

एक मेमोरी मॉडयूल से जुड़े क्रॉसबार स्विच का कार्यात्मक डिजाइन आंकड़ों में दिखाया गया है जो सर्किट में मल्टीप्लेक्स होते हैं जो मेमोरी मॉडयूल के साथ संचार के लिए एक सीपीयू से डेटा, पता और नियंत्रण का चयन करते हैं। आर्बिट्रेशन लॉजिक ने एक CPU का चयन करने के लिए प्राथमिकता स्तर स्थापित किए जब दो या अधिक CPU एक ही मेमोरी तक पहुँचने का प्रयास करते हैं। मल्टीप्लेक्सर्स को बाइनरी कोड द्वारा नियंत्रित किया जा सकता है जो मध्यस्थता तर्क के भीतर एक प्राथमिकता एनकोडर द्वारा निर्मित होता है।



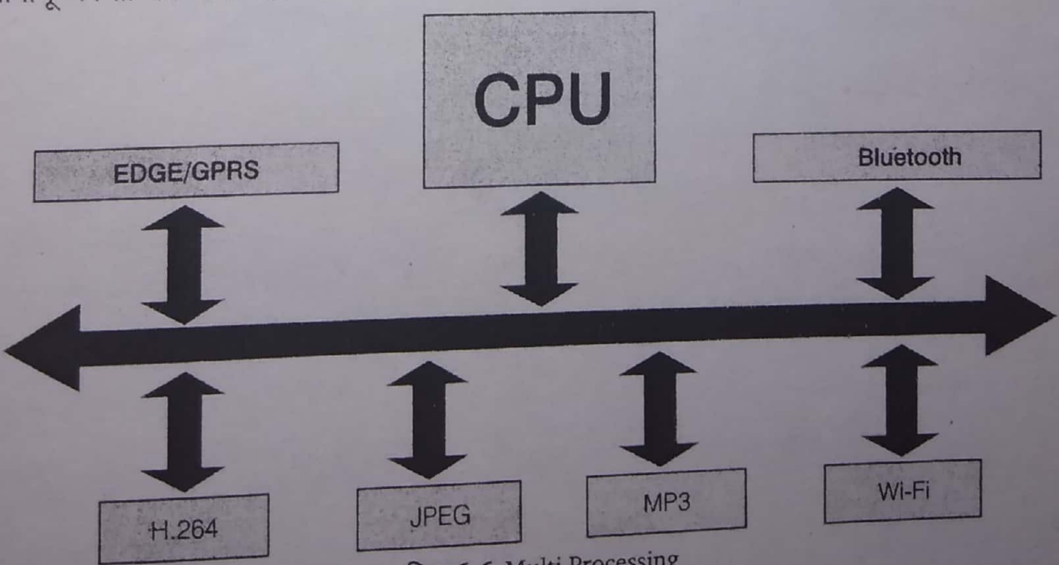
चित्र **6.5** Crossbar Switch

एक क्रॉसबार स्विच सिस्टम सभी मेमोरी मॉडयूल से एक साथ ट्रांसफर की अनुमति देता है क्योंकि प्रत्येक मॉडयूल के साथ एक अलग पथ जुड़ा हुआ है। इस प्रकार, स्विच को लागू करने के लिए आवश्यक हार्डवेयर काफी बड़ा और जटिल हो सकता है।

प्रश्न 7. मल्टीप्रोसेसिंग से आप क्या समझते हैं? मल्टीस्टेज स्विचिंग नेटवर्क को चित्र सहित समझाइए।

उत्तर मल्टीप्रोसेसिंग (Multi Processing) प्रणाली में, दो या दो से अधिक स्वतन्त्र प्रोसेसर आपस में सामंजस्य प्रणाली के तहत जुड़े होते हैं। इस प्रकार की प्रणाली में, एक या अनेक प्रोग्रामों से आने वाले निर्देश एक ही समय में अलग-अलग प्रोसेसरों के द्वारा प्रोसेस किये जा सकते हैं।

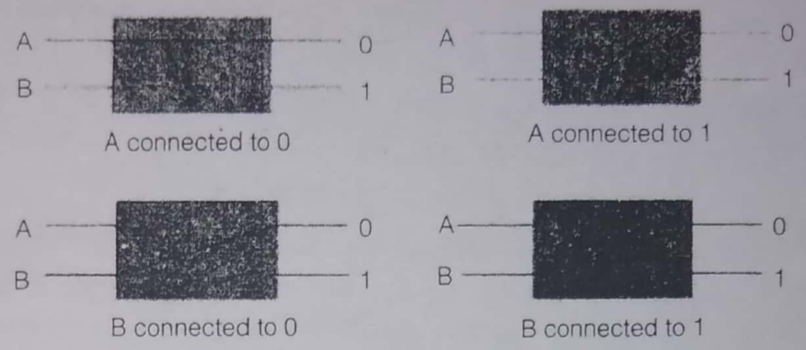
मल्टी प्रोग्रामिंग का अर्थ है एक ही सीपीयू में भागीदारी करने वाले दो या दो से अधिक प्रोग्रामों का एक ही बार में कायान्वित होना। मल्टी प्रोसेसिंग में, किसी एक समय में एक प्रोग्राम या अनेक प्रोग्रामों के विभिन्न निर्देशों को दो या दो से अधिक सीपीयू कार्यान्वित करते हैं।



चित्र **6.6** Multi Processing

मल्टी प्रोसेसिंग सिस्टम का निर्माण मल्टीप्रोसेसर सिस्टम को ध्यान में रखते हुए किया गया है। एक से अधिक प्रोसेसर उपलब्ध होने के कारण इनपुट आउटपुट एवं प्रोसेसिंग तीनों कार्यों के मध्य समन्वय रहता है। एक ही तरह के एक से अधिक सीपीयू का उपयोग करने वाले सिस्टम को सिमिट्रिक मल्टी प्रोसेसर सिस्टम कहा जाता है।

मल्टीस्टेज स्विचिंग नेटवर्क Multing Switching Network मल्टीस्टेज नेटवर्क का मूल घटक एक दो-इनपुट, दो-आउटपुट इंटरचेंज स्विच है जैसा कि चित्र में दिखाया गया है।

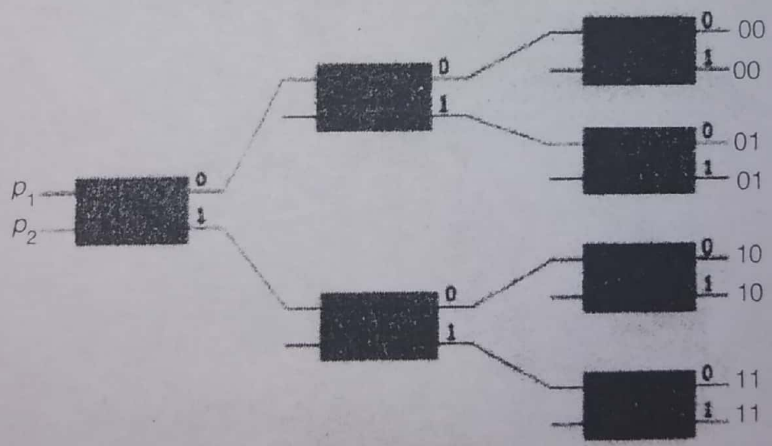


चित्र 6.7

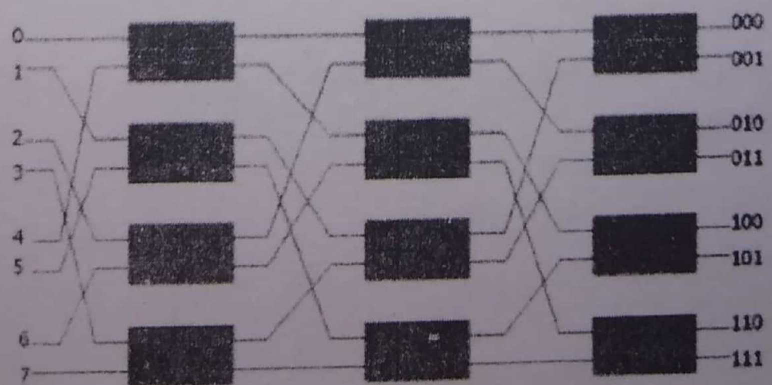
बिल्डिंग ब्लॉक के रूप में 2×2 स्विच का उपयोग करना, कई स्रोतों और गंतव्यों के बीच संचार को नियंत्रित करने के लिए मल्टीस्टेज नेटवर्क का निर्माण करना संभव है।

यह कैसे किया जाता है, यह देखने के लिए, चित्र में दिखाए गए बाइनरी ट्री पर विचार करें।

कुछ अनुरोध पैटर्न एक साथ संतुष्ट नहीं हो सकते। यानी, यदि $P1 \rightarrow 000 \sim 011$, तो $P2 \rightarrow 100 \sim 111$



चित्र 6.8



चित्र 6.9 8×8 Omega Switching Network

Scanned with CamScanner

कुछ अनुरोध पैटर्न एक साथ नहीं जोड़े जा सकते हैं। यानी, किसी भी दो स्रोतों को एक साथ 000 और 001 गंतव्य पर नहीं जोड़ा जा सकता है।

कसकर युग्मित मल्टीप्रोसेसर सिस्टम में, स्रोत एक प्रोसेसर है और गंतव्य एक मेमोरी मॉडयूल है।

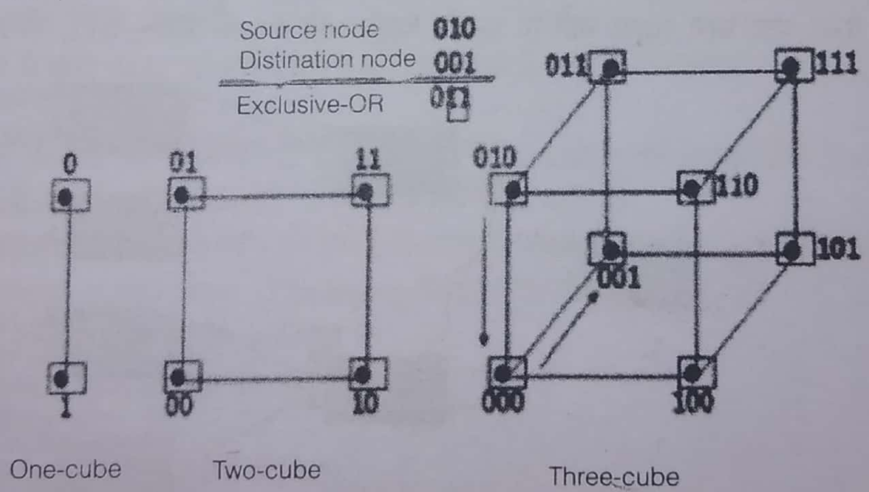
पथ सेट करें—पता को स्मृति में स्थानांतरित करें डेटा स्थानांतरित करें।

एक शिथिल युग्मित मल्टीप्रोसेसर सिस्टम में, स्रोत और गंतव्य दोनों प्रसंस्करण तत्त्व हैं।

हाइपरक्यूब प्रणाली Hypercube System हाइपरक्यूब या बाइनरी एन-क्यूब मल्टीप्रोसेसर संरचना एक शिथिल युग्मित प्रणाली है जो एन $= 2$ एन प्रोसेसर से बना है जो एन-आयामी बाइनरी क्यूब में परस्पर जुड़ा हुआ है। प्रत्येक प्रोसेसर घन का एक नोड बनाता है, जिसके प्रभाव में न केवल एक सीपीयू होता है, बल्कि स्थानीय मेमोरी और आई/ओ इंटरफेस भी होता है। प्रत्येक प्रोसेसर का पता उसके प्रत्येक n पड़ोसी से बिल्कुल भिन्न स्थिति से भिन्न होता है। चित्र में नीचे $n = 1, 2$ और 3 के लिए हाइपरक्यूब संरचना को दर्शाया गया है।

एन-क्यूब संरचनाा के माध्यम से रूटिंग संदेशों को एक नोड से एक स्रोत नोड से गंतव्य नोड तक ले जा सकता है। गंतव्य नोड पते के साथ विशेष या स्रोत नोड पते की गणना करके एक रूटिंग प्रक्रिया विकसित की जा सकती है। वह संदेश तब किसी भी एक कुल्हाड़ी के साथ भेजा जाता है, जिसके परिणामस्वरूप बाइनरी मूल्य में कुल्हाड़ियों के समान 1 बिट्स होंगे, जिस पर दो नोड्स भिन्न होते हैं। हाइपरक्यूब आर्किटेक्चर का एक प्रतिनिधि इंटेल आईपीएससी कम्प्यूटर कॉम्प्लेक्स है।

इसमें 128 ($n = 7$) माइक्रो कम्प्यूटर शामिल हैं, प्रत्येक नोड में एक सीपीयू,एक फ्लोटिंगपॉइंट प्रोसेसर, स्थानीय मेमोरी और सीरियल संचार इंटरफेस इकाइयाँ शामिल हैं।



चित्र 6.10 Hypercube Structures for $n = 1, 2, 3$

मॉडल प्रश्न-पत्र

कम्प्यूटर आर्किटेक्चर और हार्डवेयर रख-रखाव

समय : 2.30 घण्टे] [पूर्णांक

नोट—

(i) सभी प्रश्नों को हल कीजिए।

(ii) विद्यार्थियों द्वारा पेज़र अथवा मोबाइल फोन का उपयोग अनुमन्य नहीं है।

1. निम्नलिखित में से किन्हीं दो भागों के उत्तर दीजिए— [2 × 5 =

(अ) रजिस्टर ऑर्गेनाइजेशन से आप क्या समझते हैं? इसके प्रकारों का वर्णन कीजिए।

(ब) पुश फंक्शन (Push Function) एवं पॉप फंक्शन (Pop Function) के बारे में समझाइए।

(स) तीन एड्रैस निर्देश, दो एड्रैस निर्देश तथा शून्य एड्रैस निर्देश को समझाइए।

2. निम्नलिखित में से किन्हीं दो भागों के उत्तर दीजिए— [2 × 5 =

(अ) इफेक्टिव एड्रैस (Effective Address) क्या होता है? इसके प्रकारों का वर्णन कीजिए।

(ब) CISC क्या है? CISC एवं RISC में तुलना कीजिए।

(स) मेमोरी अनुक्रम (Memory Hierarchy) को चित्र सहित समझाइए।

3. निम्नलिखित में से किन्हीं दो भागों के उत्तर दीजिए— [2 × 5 =

(अ) सीपीयू (CPU) में मेमोरी कनेक्शन को चित्र सहित समझाइए।

(ब) कैशे मेमोरी (Cache Memory) का वर्णन कीजिए।

(स) हार्डवेयर एल्गोरिथ्म (Hardware Algorithm) को समझाइए।

4. निम्नलिखित में से किन्हीं दो भागों के उत्तर दीजिए— [2 × 5 =

(अ) भाग विधि की हार्डवेयर एल्गोरिथ्म विधि का फ्लोचार्ट सहित वर्णन कीजिए।

(ब) डाटा ट्रांसफर विधियों का वर्णन कीजिए।

(स) माइक्रोप्रोसेसर 8085 से आप क्या समझते हैं? चित्र बनाकर दर्शाइए।

5. निम्नलिखित में से किन्हीं दो भागों के उत्तर दीजिए— [2 × 5 =

(अ) माइक्रोप्रोसेसर 8085 में पिन से सम्बन्धित फंक्शन एवं सिगनल का विवरण दीजिए।

(ब) पैरेलल प्रक्रिया (Parallel Processing) क्या है? उदाहरण सहित बताइए। इसके लाभ लिखिए।

(स) क्रॉसबार (Crossbar Switch) का संक्षेप में वर्णन कीजिए।

Scanned with CamScanner