16/1/23

Computer Architecture and system

The history of computer

- From compute → computer. What is computer? what's the characters
- The father of the engineer computer = Charles Abes. (Abbacus). Chard's babaj
- i) The 1st electronic computer / 1st generation (mechanical era) computer is vaccuam tube. [1946 onwards] [language used at that time = Assembly language, low level language, no use of high level language]
  Example = univac, Ivac. [Disadvanged = very large, takes much space]
- ii) After that transistor comes. It is very small, take less space. High level language is used → 2nd generation.
- iii) 3rd generation, IC used. Introduced after 1967. After that multiprogramming concept came. In small scale integration = 1 chip has 1000 IC, Large " " = 1 chip = 10,000 20,000 IC etc. [Representative machine = IBM, windows]
- iv) 4th generation = semiconductor memory. Real time OS = the os that is used in PC. (1975-1974)
- Distributive Programming = A computer program that runs within a distributed system is called a distributed programs.

• Parallel Programming: In simple terms, it is the process of decomposing a problem into smaller tasks that can be executed at the same time using multiple compute resources.

• Centralized Programming: It is a type of computing architecture where all or most of the processing/computing is performed on a central server. [cloud.]

• 1st Programmer is a female = Ada Lovelace (from London) (1815-1852)

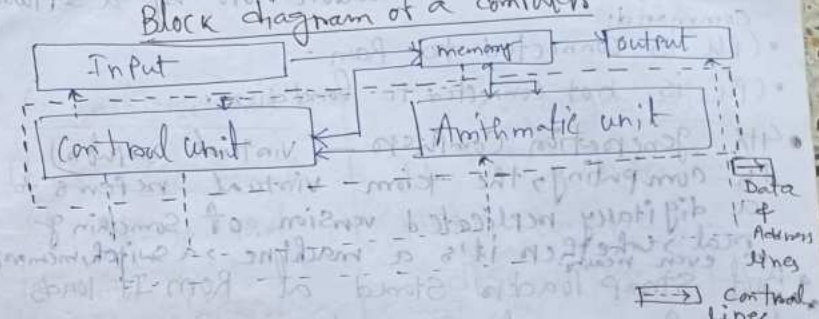
• Store Programme Concept (JAS computer): All the Prog and datas are stored in the same memory (idea of christen [A computer that stores instructions in its memory to enable it to perform a variety of tasks in sequence or intermittently]

• Hardware architecture — hardware architecture is the representation of an engineered (or to be engineered) electronic or electromechanical hardware system, and the process and discipline for effectively implementing the design(s)

• Christian university architecture — for such a system. The christian colleges included here offer your major of interest you

Von neumann architecture — in PPT (2022)

Block diagram of a computer



Computer system contains 3 resources
1) Hardware 2) Software 3) Human resources
keyboard, mouse (peripheral devices)

CPU — has 2 components 1) CU 2) ALU

CU → Access the brain of the computer, give instructions to do work

ALU → It performs all the operations

• Bringing data memory → CPU ⇒ load
• Storing data memory ← CPU ⇒ store
Address movement is one direction
take input is a controle instruction
output

BIOS = Basic Input Output System (F2/del)
POST = Power on self test (is a test whenever you are switing on PC)

Booting = In computing, booting is the process of starting a computer as initiated via hard-

Harvard Architecture - CPU cannot do both things together (Read the instruction & read (write data). Harvard Architecture is the computer architecture that contains separate storage & separate buses (Single path) for instruction & data. It was basically developed to overcome the bottle neck of von neumann's Architecture. The main advantage of having separate buses for instruction and data is that the CPU can access instructions & read/write data at same time.

ware such as a button or by a software command.

- CPU is connected to Ram.
- CPU is not connected to hard disk.
- 4th generation Computers — (Virtual computers) In computing, the term virtual refers to a digitally replicated version of something real, whether it's a machine, a switch, memory or even network.
- Boot Strap loaders Stored at Rom. It loads OS from hard disk to Ram (booting).

- Compilers, OS → System Programm
- Powerpoint, Excel, word etc → Application Programm

what is Computer organization & Architecture.

(Transparent to the Programmer) Ex- Interfacing

(visiable parts of the computer)

20/1/23

C. architecture

It is the branch of study that is with the conceptual design & basic overviews of a computer system. It refferes to the parameters of a computer system these are visiable to the users. It deals with the attributes that have a direct impact on the execution of a programme. It includes instruction addressing modes, instruction formats, instruction sets etc.

C. organization

It is a lower level more concrete discription of the system that involves how the constituents of part of the system are interconnected and how they interoperate in order to realize the architecture specifications. The organisational issues include the hardware details transparent for the Programmers like the memory technology, the interfacing between CPU and the Peripherals, system interconnect components like Computer

buses and switches.
EX —

- What is CPU, key board etc ——— (Basic things)

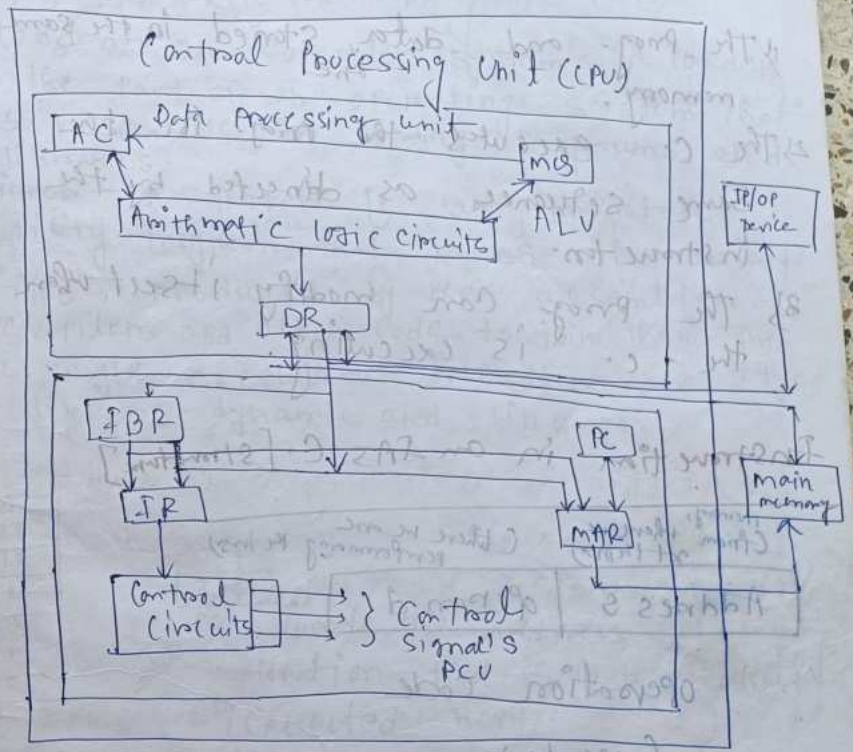
CPU — A central processing unit, also called a central processor, main processor or just processor, is the electronic circuitry device that executes instructions comprising a computer program. The CPU performs basic arithmetic, logic, controlling, and input/output operations specified by the instructions in the program.

Keyboard A computer keyboard is a peripheral input device modeled after the typewriter keyboard. which uses an arrangement of buttons or keys to act as mechanical levers or electronic switches.

monitor A computer monitor is an output device that displays information in pictorial or textual form.

mouse — A computer mouse is a hand-held pointing device that detects two-dimensional motion relative to a surface. the motion is typically translated into the motion of a pointer on a display, which allows a smooth control of the graphical user interface of a computer.

- Structure of an IAS computer

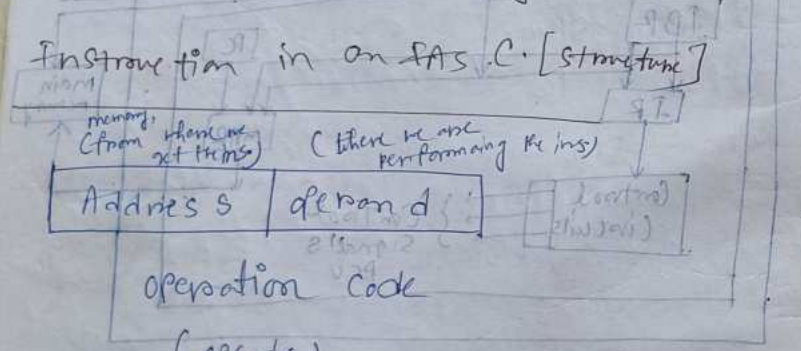


PCU: Program control unit
AC: Accumulator
MQ: miltiplier-quotient
DR: Data Register
IBR: Instruction Buffer Register
IR: Instruction Register
MAR: memory Address Register
PC: program counter

[Designed-1946]
All the registers are invented by vaccum tubes

Features of IAS Computer

1) The prog. and data stored in the same memory.
2) The C. executes the prog. in the same sequence as directed by the instruction.
3) The prog can modify itself when the C. is executing.



- the size of register = 40 bits.
- Per memory contains storage of 40 bits.
- CPU part is controlled by a clock.
- Fetching → decoding → operation

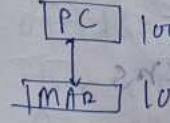
AC = Source of the operand and destinating of operand is AC.
the result of any operation

It is most imp register.

- AC, MQ = used for storing the result.
- loader = In computer systems a loader is the part of an operating system that is responsible for loading programs and libraries.
- linker — A linker is an important utility program that takes the object files, produced by the assembler and compilers, and other code to join them into a single executable file. there are 2 types of linkers, dynamic and linkage.

1000
PC = loads the address of memory location that is to be fetched (executed next.

MAR = the prog. that is getting to be fetched (work of MAR)



if we have a loop then PC > MAR may not be followed.

type of qn

(CS) (1) Explain the IAS structure?
(2) What is Store Prog. Concept?
(3) Diff between Human and Computer and Hardware and Software? (may be Christ or University)

④ where hardware arch. is used?
⑤ what is Von-Neuman Bottleneck?
⑥ what is the remedy of Von-Neuman Bottleneck?

• Embedded system- An embedded system is a computer system - a combination of a computer processor, computer memory and i/p/o/p peripheral devices - that has a dedicated function within a larger mechanical or electronic system.
• frequently executed prog. are used cache memory.
• ~~for~~ C.a. is used for general use of comp.; h.c is used for mobile
[C.a = Chirston University comp. arch.]
[h.c = Harvard " "]

Von-Neumann Bottleneck

~~The speed difference between~~
It is a limitation on throughput caused by the standard personal computer architecture. the term is named for John von Neumann, who developed the theory behind the architecture of modern computers. Earlier computers were fed programs and data for processing while they were running.

① Explain the IAS structure.

* the diagram.

The IAS machine was a binary computer with a 40-bit word, storing two 20-bit instructions in each word. the memory was 1024 words (5.1 kilobytes). Negative no. were represented in two's complement format. It had two general-purpose registers available: the accumulator (AC) and multiplier/quotient (MQ).

② what is stored program concept?

Stored-program computers, a computer that stores instructions in its memory to enable it to perform a variety of tasks in sequence or intermittently. One of the most imp feature of a stored program computer is being able to store the programs as well as the data together in the memory. This concept of the stored-program computer can be traced back to the 1936 theoretical concept of a universal turing machine. The 5 essentials of stored prog concepts are - an arithmatic logic unit, a control unit, a memory, some form of input/output and a system bus that provides a data path between these parts.

③ Difference between von Neumann and harvard architecture –

von Neumann Arch.	Harvard arch.
1) It is ancient computer architecture based on stored program computers concept.	1) It is modern computer architecture based on Harvard mark I relay based model.
2) Same physical memory address is used for instructions and data.	2) separate physical memory address is used for instructions and data.
3) There is common bus for data and instruction transfer.	3) seperate buses are used for transferring data and instruction.
4) two clock cycles are required to execute single instruction.	4) An instruction is executed in a single cycle.
5) It is cheaper in cost.	5) CPU can access instructions and read/write at the same time.
6) It is used in personal computers and small computers.	6) It is used in micro controllers and signal processing.

④ where harvard Architecture is used?

It is used primary for small embedded computers and signal processing (DSP). Von Neumann is better for desktop computers, laptops, workstations and high performance computers. some computers may use advantages from both architectures.

⑤ what is von Numann bottleneck?

done.

⑥ what is the remedy of von Numann bottleneck?

caching is a common method used to overcome the Von Neumann bottleneck. Prefetching. Instructions and data that are expected to be used most are fetched into the cache in advance so they're immediately available when needed.

they are

24/1/23

COA

Data Representation of Computer Architecture

> Have to study the entire number system, complement, signed magnitude representation, range of signed mag, Disadvantages of signed mag, 1's complement, 2's complement, advantages, fixed point representation, arithmetic operating of fixed point no, 10's complement. (maam is skipping)

Floating point representation ***

32×10^{5} → 5 → exponent; 32 → mantissa; 10 → Base

-101×2^{100}, Base = 2

Mantissa and exponent will be stored in memory, no need to store base

IEEE 754

sign	Exponent	mantissa
← 1 →	← 8 →	← 23 →

← 32 →

The format for representing floating point No. = $\pm M \times B^{\pm E}$

→ there is no space of storing the sign of exponent. that's why Biased exponent is needed.

range of 8-bit no - [-127 to 128] $[-2^{n-1}-1 \text{ to } 2^{n-1}]$

Biased exponent we will add the +ve no. of exponent to make of 0, that's why the exp come to 0 → 255 [For 8 bit]

Ex = 5×10^{-127} add 127

$= 5 \times 10^{-127+127}$

$= 5 \times 10^{0}$

Normalization [formula = $(1.N)\, 2^{E-127}$]

111.01×2^{3}

↓ normalization

1.1101×2^{7} [one digit atmost have to present before point] (implicit)

sign	exponent	1101000...0
		mantise

Imp

① Represent -1.75 as IEEE 754 floating point format

Soln

2 | 1 − 1 (01)

0.75 × 2 = 0.5 — carry 1

0.5 × 2 = 0 — carry 1

$-1.75 = (01.11) \times 2^{0}$ } any of that form (→ implicit point)

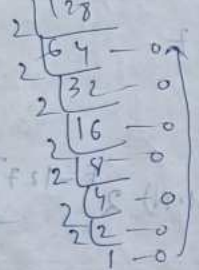
$= -0.111 \times 2^{1}$ (mantisa format)

a) Decimal number -1.75 is represented as: $-11.11_{(2)} \times 2^0$ (hidden bit)

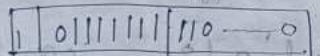
b) Normalization form = -0.1111×2^1

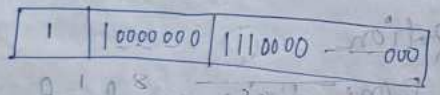
c) Bias $\rightarrow 2^{K-1} - 1 = 2^{8-1} - 1 = 127$

Biased exponent = $1 + 127$
= 128



Biased exponent
= $0 + 127$
= 127





② A floating point no used 16 bits for representating a no. The most significant bit is the sign bit. The least significant 9 bits represent the mantisa and the remaing 6 bits represent the exponent. Assume that the no are stored in a normalized format with one hidden bit. a) Find the representation of -1.6×10^3 in this no. System?

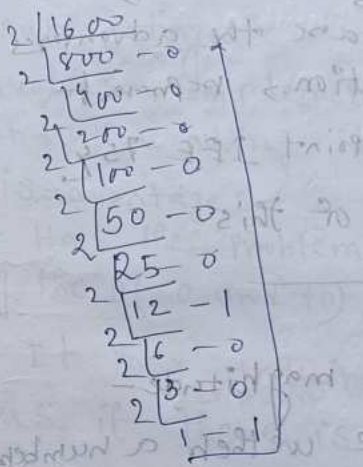
b) What is the value represented by 0000|0011000000000

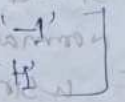
Sol - a) Decimal number -1.6×10^3 is represented as

binary: $-1100|1000000000$

b) Normalization form = $-1.1001000000000 \times 2^{10}$
= $-1.1001000000000 \times 2^{10}$

-1.6×10^3
= -1600





c) Bias $\rightarrow 2^{K-1} - 1 = 2^6 - 1 = 31$

Biased exp = $10 + 31 = 41$

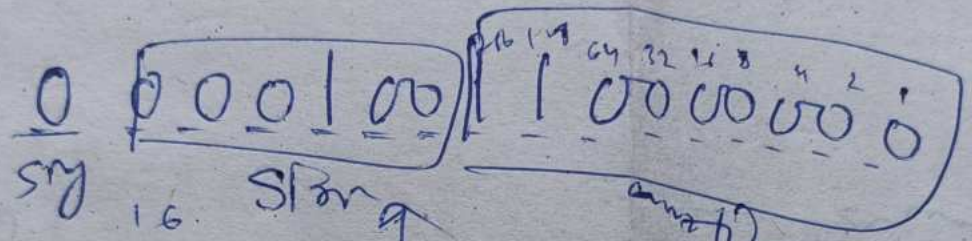
b) 0000|100|1000000000

4 (exp) mantisa

exp' = $4 - 31 = -26$

after normalization there will be a hidden 1 = 1.1000000000

Exp + 31 = 4
⇒ Exp = 4 - 31
Exp - 31 = Exp (actual exp)
⇒ 4 - 31 = x
⇒ x = -26



C'

$(1-N) \times 2^{E-31}$ → IEEE 754

E − 31 ← = 4 − 31 = −26

31

1.384×10^{-26}

EXP + 31 = 4

If exp is big then represent it in exp form while representing IEEE 754
Don't use 2 exp form

~~Binary~~

by $+1.384 \times 10^{-26}$

① Biased representating, normalization, what are the advantages of using normalization, represent a no. in floating point IEE 754. Biased exp
represent the reverse of this.

• Fixed point representations
Number System

① Advantages of singed maghitude–
you can determine wether a number is negative or non negative simply by testing the most significant bit
It is easy to understand and encode.
Dis advantage– one of the bit patterns is wasted. Addition doesn't work the way we want it to do.
1) Need additional bit for sign.
2) Not convenient for arithmetic evaluation
3) They are two representations of zero.

: 00000000 = $+0_{10}$
: 10000000 = -0_{10}

② 1's Complement Representation–
Advantages– i) still relatively simple to represent the numbers.
ii) Simpler Add/Subtract circuit design (subtracting a number from another involves complementing the subtracted and then adding it to the other number).
Disadvantages–
i) Has the problem of double representing the 0 (-0 and +0),
ii) It may require two addition operations as, if there is a carry out it has to be added to get the correct result.

③ Advantages and Disadvantages of 2's Complement–
Advantages– i) only one representation of zero
ii) Addition algorithm will not depend on the "sign" of a number.

Disadvantages– i) one more negative number than there is positive number.

-32 = 10000, but there is no way to show +32. (No way to show the representation of (+)ve no.)

④ overflow (underflow) –

Definition of overflow –

Result of an arithmetic operation is too large (or small) to be represented in the number of bits available. / The answer to an addition or subtraction exceeds the magnitude that can be represented with the allocated no. of bits.

Detection – varies with the representation.

For unsigned binary, this is determined by a carry out of the MSB.

```
 ①
   10101101
 + 01101100   8 bit operands
 ----------
  100011001
```

⑤ Carry and overflow –

Is this an overflow condition –

```
   11101100
   10101101
 + 01101100
 ----------
  100011001
```

If this is a two's complement number – No!

A negative number plus a positive number cannot produce overflow on a two's complement architecture.

The range of signed magnitude

⑥ For unsigned Binary Number: 0 to $(2^n - 1)$

$\rightarrow 0$ to $(2^8 - 1) \rightarrow 0$ to 255

For signed Binary Number:

$-(2^{n-1} - 1)$ to $(2^{n-1} - 1)$

$-(2^7 - 1)$ to $(2^7 - 1) \rightarrow (-127)$ to 0 to $(+127)$

⑦ The range of 1's complement – the range of 1's complement form is from $-(2^{n-1} - 1)$ to $(2^{n-1} - 1)$, i.e. Range of 6 bit 1's complement form binary number is from $-(2^5 - 1)$ to $(2^5 - 1)$ which is minimum −31 (i.e. 100000) to maximum value +31 (i.e. 011111). And Zero (0) has two representation −0 (i.e. 111111) and +0 (i.e. 000000).

⑧ Range of 2's complement –

In general, the range of an N-bit two's complement number spans $[-2^{N-1}, 2^{N-1} - 1]$. It should make sense that there is one more negative number than positive number because there is no −0.

(9) fixed point representation - In computers, fixed-point representation is a real data type for numbers. Fixed point representation can convert data into binary form, and then the data is processed stored, and used by the computer. It has a fixed number of bits for the integral and fractional parts.

(10) 10's complement - 10's complement of a decimal number can be found by adding 1 to the 9's complement of that decimal number. It is just like 2's complement in binary number representation. Eg: the decimal no. 456, 9's complement of this no. will be 999-456 which will be 543. Now, 10's complement will be 543+1 = 544.

(11) Normalization - It is the process of reorganizing data in a database so that it meets two basic requirements: there is no redundancy of data, all data is stored in only one place; Data dependencies are logical, all related data items are stored together.

(12) Bias representation - offset binary, also referred to as excess-K, excess-N, excess-e, excess code or biased representation, is a method for signed numbers representation where a signed number n is represented by the bit pattern corresponding to the unsigned number n+K, K being the biasing value or offset.

(13) Exponent bias - In floating-point arithmetic, a biased exponent is the result of adding some constant (called the bias) to the exponent (chosen to make the range of the exponent nonnegative. Biased exponents are particularly useful when encoding and decoding the floating-point representation

of subnormal numbers.

(14) Advantages of normalization →

1) Direct translation from logical to physical design in RDMS.

2) Reduced data redundancy.

3) Protection against update and delete anomalies.

4) Ability to add or delete entities, ~~all~~ attributes and relations without wholesale restructuring of tabels.

Disadvantages of Normalization →

1) maintaining more tables is a bit messier.

2) Nested queries over multiple tables gets tricky.

(15) ~~Advantages of~~

(13)

Addition & Substraction (Binary)

Half adder - It is a combinational logic circuit. You can design it by connecting one AND gate and one xor gate. A half-adder circuit consists of 2 input terminals namely A and B. Both of these add two input digits (one-bit numbers) and generate the output in the form of a carry and a sum.

x + y	0 + 0	0 + 1	1 + 0	1 + 1
C S	0 0	0 1	0 1	1 0
	C S	C S	C S	C S

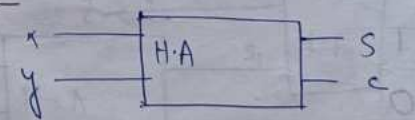
Carry Sum

x	y	MSB C	LSB S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

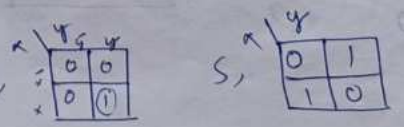
← T.T

$C = x \cdot y$, $S = x \oplus y$ ← Eqn

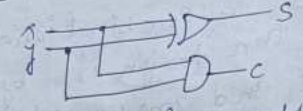
Symbol



K-maps



circuit -



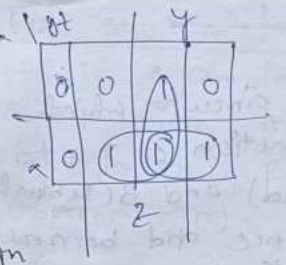
- Full Adder – A full adder circuit is central to most digital circuits that perform addition or subtraction. It is so called because it adds together two binary digits, plus a carry-in digit to produce a sum and carry-out digit. It therefore has three inputs and two outputs.

x	0	0	0	0	1	1	1
+ y	+ 0	+ 0	+ 1	+ 1	+ 0	+ 1	+ 1
+ z (carry in)	+ 0	+ 1	+ 0	+ 1	+ 0	+ 1	+ 1
	00	01	01	10	01	10	11
	C S						
	carry out	Sum					

TT

x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

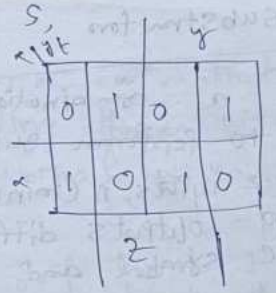
C =



Eqn

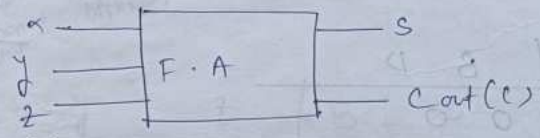
$C = xz + xy + yz$

S =

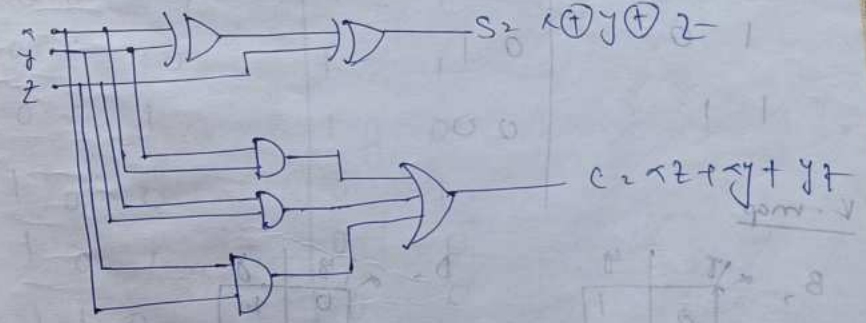


$S = x \oplus y \oplus z$

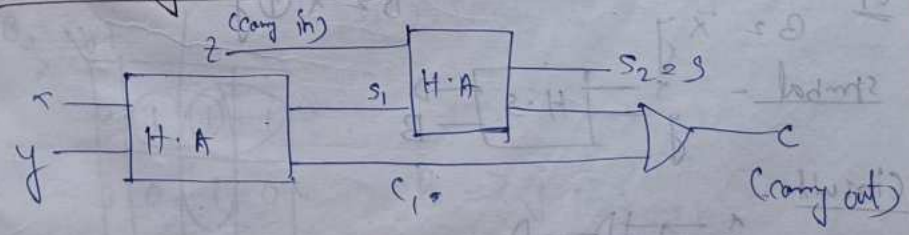
Symbol



Circuit



- F.A using H.A



• Half Substractor

It is a combinational circuit which is used to perform subtraction of 2 bits. It has 2 inputs, x (minuend) and B (subtrahend) and two outputs difference and borrow. The logic symbol and truth table are

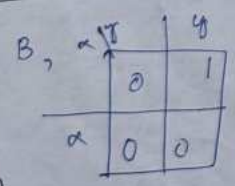
x	0	0	1	1
-y	-0	-1	-0	-1
B D	00	11	01	00

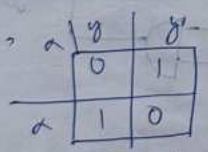
B → Borrow, D → Difference

TT

x	y	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

K-map



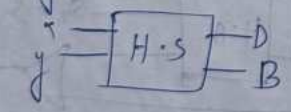


Eqn

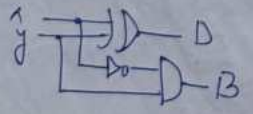
B = $\bar{x}y$

D = $x \oplus y$

Symbol -



Circuit -



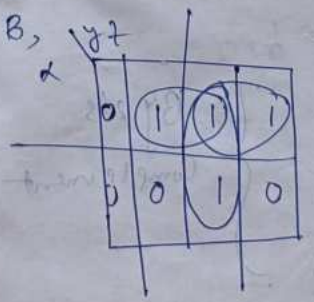
• Full Substractor

It is a combinational ckt. that performs subtraction involving three bits, namely A (minuend), B (subtrahend) and B in (borrow-in).

x	0	0	0	1	1	1	1
-y	-0	-1	-0	-1	-0	-1	-0	-1
-z (Borrow in)	00	11	11	10	01	00	00	11

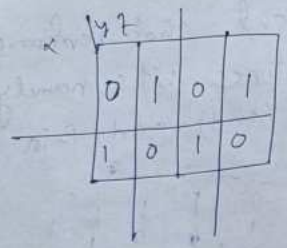
B → Borrow out, D → Difference

x	y	z	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



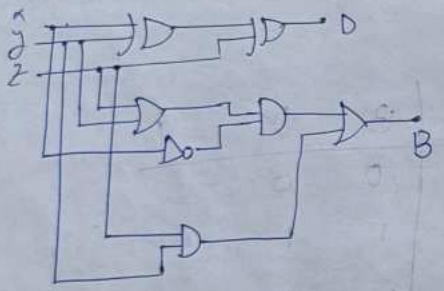
$B = \bar{x}y + \bar{x}z + yz$

$= \bar{x}(y+z) + yz$

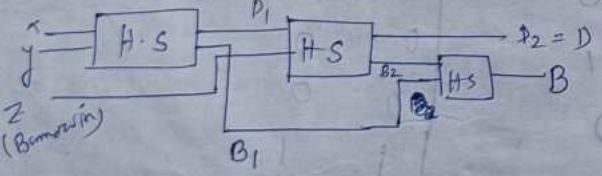


$D = x \oplus y \oplus z$

• Circuit



• FS in the form of H.S.



• Subtraction (by using 4 bit)

$\begin{array}{r} 7 \\ -5 \\ \hline 2 \end{array} \rightarrow \begin{array}{r} 0111 \\ +1011 \\ \hline 10010 \end{array}$ (+2)

$\begin{array}{r} 5 \\ -7 \\ \hline -2 \end{array} \begin{array}{r} 0101 \\ +1001 \\ \hline 1110 \end{array}$ 2's → −0010 (−2)

0101 → 2's → 1011

0111 → 1001

By 2's Complement

$\begin{array}{r} 7 \\ -5 \\ \hline 2 \end{array} \begin{array}{r} 0111 \\ +1010 \\ \hline 10001 \end{array}$ (+2)

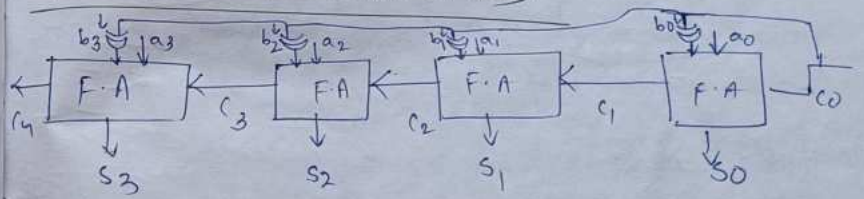
0101 → 1010

$\begin{array}{r} 5 \\ -7 \\ \hline -2 \end{array} \begin{array}{r} 0101 \\ +1000 \\ \hline 1101 \end{array}$ 1's → −0010 (−2)

0111 → 1000

1's Complement

Adder / subtractor (A ± B)



Bitwise complement (B)

1's complement +1

$\begin{array}{r} 2 \\ +3 \end{array} \rightarrow \begin{array}{r} 0010 \\ 0011 \\ \hline 0101 \end{array}$ $C_0 = 0$ | If $C_0 = 0$ pass B and it will be simble addition

$\begin{array}{r} 2 \\ -3 \end{array} \rightarrow \begin{array}{r} 0010 \\ +1100 \\ \hline 11101 \end{array}$ $C_0 = 1$ | If $C_0 = 1$ pass complement of B and then it is normal addition

2's complement

−0001 (−1)

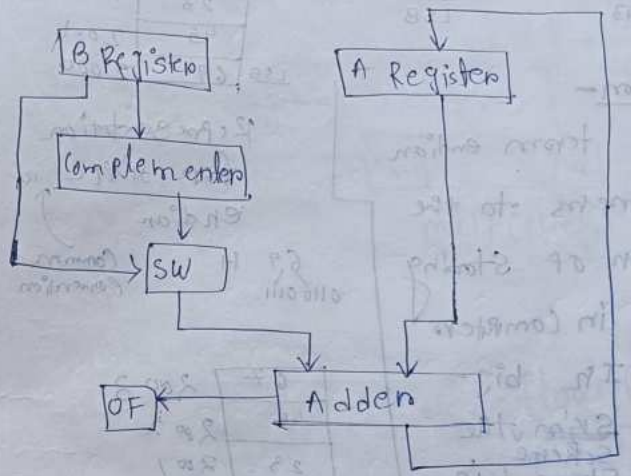
CW1

30/1/23

Hardware implementation for unsigned no.

1) Adder, Subtractor (Addition, Subtraction)

Done

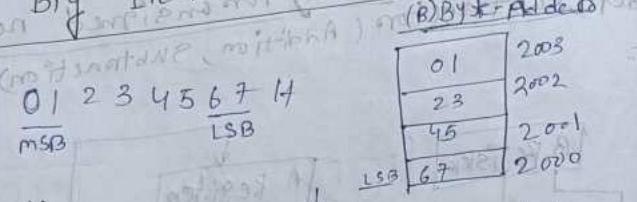


OF = Overflow bit

SW = Switch (select addition or subtraction)

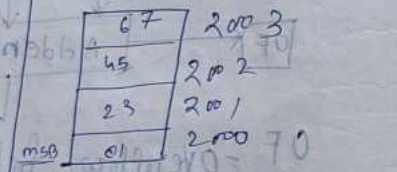
Block diagram of ~~hardware~~ for Addition and subtraction

Big Endian & Little Endian



Representation for little endian

Common convention

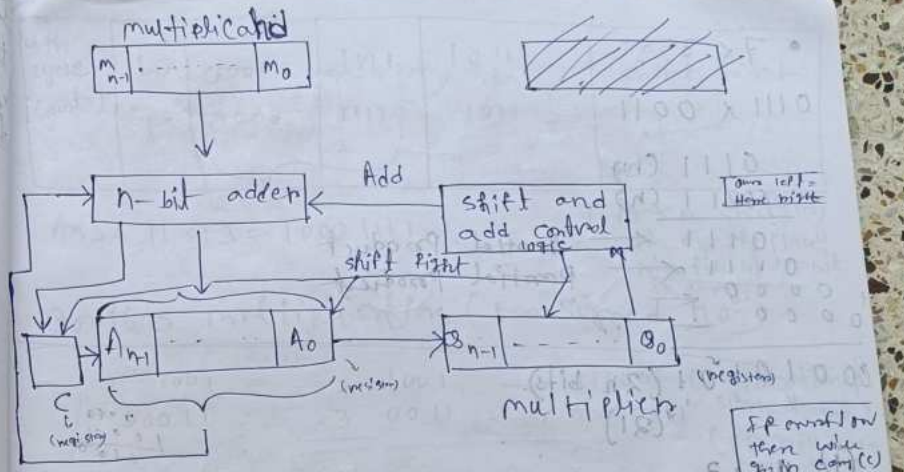


Big Endian representation

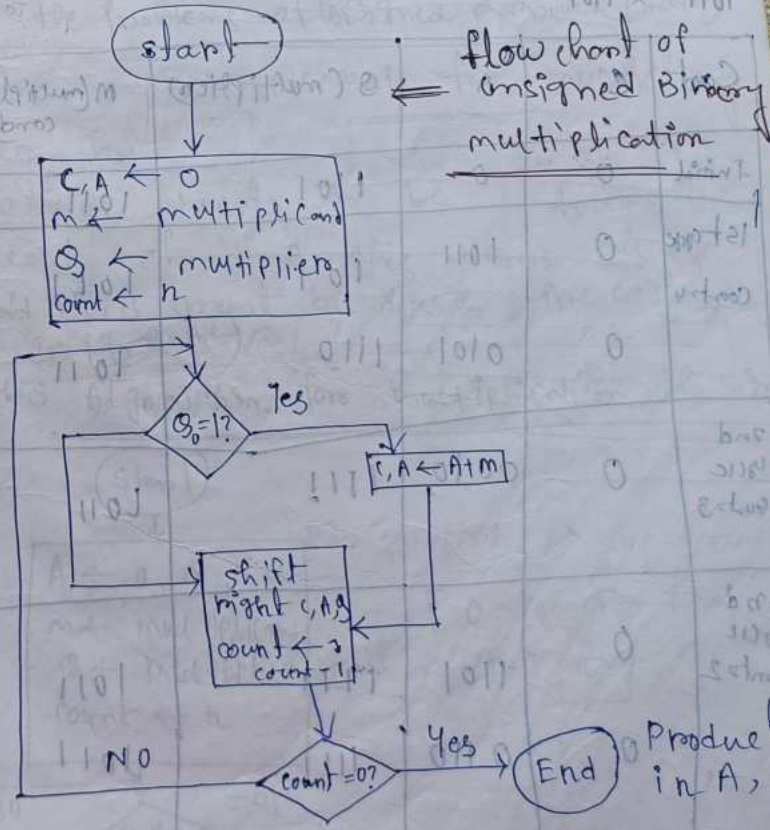
Defination-

The term endian refers to the order of storing bytes in computer memory. In big-endian scheme the MSB is stored in the lowest memory address and the while in little endian scheme the LSB is stored in the lowest memory address.

multiplication of unsigned Numbers



flow chart of unsigned Binary multiplication



• 7 × 3

0111 × 0011

```
   0111 (m)
 × 0011 (n)
 --------
   0111   ← Partial product
  0111    ← Partial product
+0000     ←
0000      ←
---------
00010101 (2n bits)
         (21)
```

① 11 × 13

1011 × 1101

Count	C	A	Q (multiplier)	M (multiplicand)	
Initial	0	0	1101	1011	
1st cycle count=4	0	1011	1101	1011	A ← A+M
	0	0101	1110	1011	shift right C,A,Q
2nd cycle count=3	0	0010	1111	1011	shift right C,A,Q
3rd cycle count=2	0	1101	1111	1011	A ← A+M
	0	0110	1111	01011	shift right C,A,Q
4th cycle count=1	1	0001	1111	1011	Add
	1	1000	1111	1011	shift right

Ans = 11 × 13 = 1000 1111

[After shifting the empty place will fill by the bit in carry]

Booth's multiplication (For signed no.)

```
 -7  1001  → 11111001  (replication of signed bit)
×3   0011  → 00000011
```

(it will make 4 bit → 8 bit, 8 bit → 12 bit)

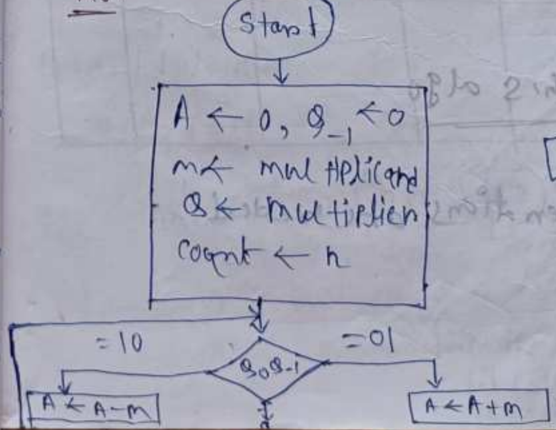
~~Advan~~ Imp The problems of unsigned approach (multip.)

i) when very inefficient for multiplying neg. no.

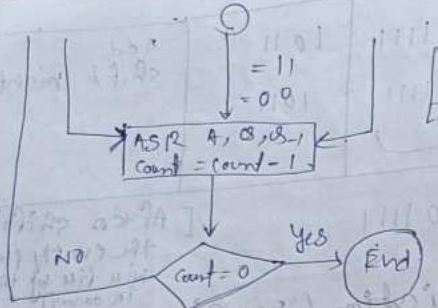
ii) Precession needs to be doubled ahead of time. This means all the additions must be double precession.

Imp Imp (Imp - All)

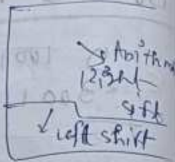
• Booth's Algorithm for multiplication of signed no.

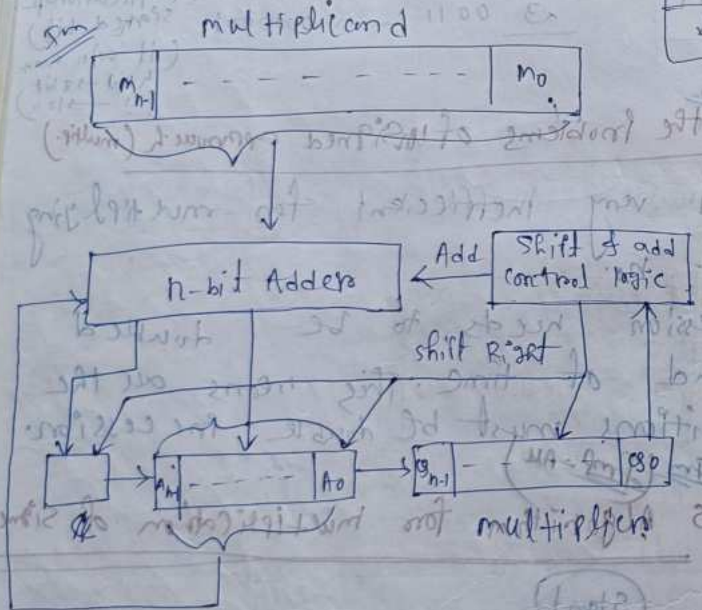


[0 = connector for P.T.O]



[ASR = Arithmatic shift Right]





Advantage of Booth's algo

0111110
only 2 operations are needed

Ex-1

7 x 3

1000 → (1)100 preserving for sign bit

A = 0000
− 0111
1001

0000
+ 1001
1001

	A	Q (multiplier)	Q_{-1}	M (multiplicand)	
	0000	0011	0	0111	Initial value
1st cycle	1001	0011	0	0111	A ← A − M
	1100	1001	1	0111	through ASR A, Q, Q_{-1}
2nd cycle	1110	0100	1	0111	ASR, A, Q, Q_{-1}
3rd cycle	0101	0100	1	0111	A ← A + M
	0010	1010	0	0111	ASR, A, Q, Q_{-1}
4th cycle	0001	0101	0	0111	ASR, A, Q, Q_{-1}

1110
+ 0111
(1)0101

Ans = 00010101 $(21)_{10}$

[what is shifting will fulfill the empty place]

[i.e. the 1st bit will repeat twice] ← [the empty bit will filled by the 1st bit]

HW

① -7 × 3 (multiply it)

1001 × 0011

	A	Q (multiplier)	Q_{-1}	M (multiplicand)	
Initial	0000	0011	0	1001	Initial values
1st cycle	0111	0011	0	1001	A ← A − m
	0011	1001	1	1001	ASR, A, Q, Q_{-1}
2nd cycle	0001	1100	1	1001	ASR, A, Q, Q_{-1}
3rd cycle	1010	1100	1	1001	A ← A + m
	1101	0110	0	1001	ASR, A, Q, Q_{-1}
4th cycle	1110	1011	0	1001	ASR, A, Q, Q_{-1}

$\therefore$ ans = $|1101011| = (-21)_{10}$

21 = 00010101

↓ 2's

11101011 (−21)

Booth's Algorithm It is a multiplication algo. that multiplies 2 signed binary numbers in two's complement notation. The algo. was invented by Andrew Donald Booth in 1950.

Im

Advantages of Booth's Algo

i) The algorithm can handle positive & negative multiplier uniformly.

ii) It achieves efficiency in the no. of additions required when the multiplier has a large block of ones or a large block of zeros.

iii) [1000 01 11 0 — 3 times add/subtract]

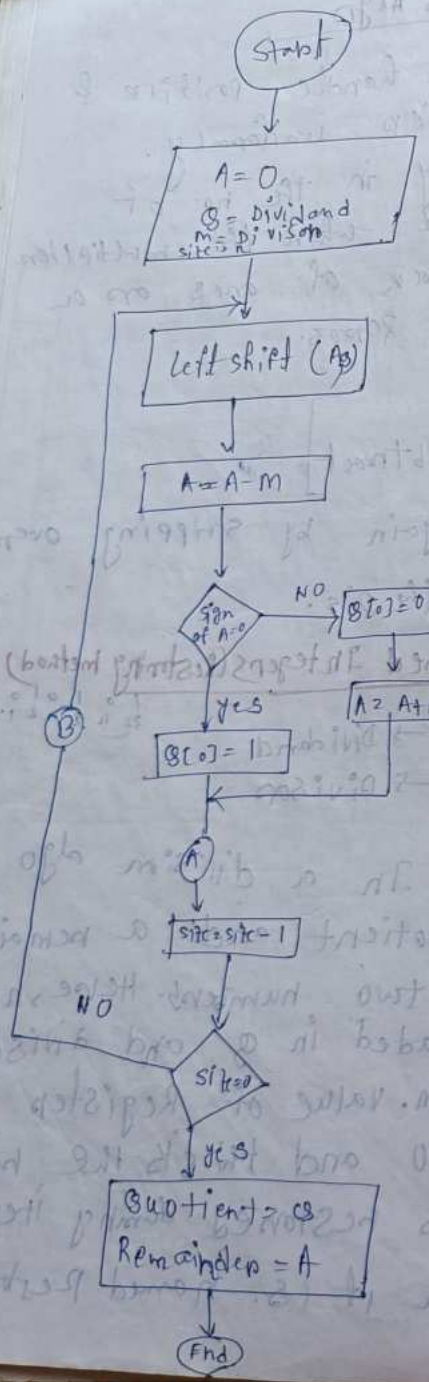
iv) The speed is gain by skipping over the 1's and the 0's.

Division of unsigned Integers (Restoring method)

e.g. D = 42 = 101010 → Dividend

D = 11 = 1011 → Divisor

Restoring method (Algo.) – In a division algo there is a quotient and a remainder when we divide two numbers. Here, n-bit dividend is loaded in Q and divisor is loaded in M. value of Register A is initially kept 0 and this is the register whose value is restored during iteration due to which it is named Restoring.



Ex-0

7 ÷ 3

size	m (Divisor)	A	Q (Dividend)	
	0011	0000	0111	Initial values
4	0011	0000	111_	Left shift A, Q
	0011	1101	1110	A ← A - m
	0011	0000	1110	since A < 0, so, Q[0] = 0 and A = A + m
3	0011	0001	110_	Left shift A, Q
	0011	1110		A ← A - m
	0011	0000	1100	since A < 0, so, Q[0] = 0 and A = A + m
2	0011	0001	100_	Left shift A, Q
	0011	0000		A ← A - m
	0011	0000	1001	since A < 0, so, Q[0] = 1 and A = A + m
1	0011	0001	001_	Left shift A, Q
	0011	1110		A ← A - m
	0011	0001	0010	A ≥ 0, Q[0] = 0 and A = A + m

Q (Quotient) = 0010 $_{(2)_{10}}$

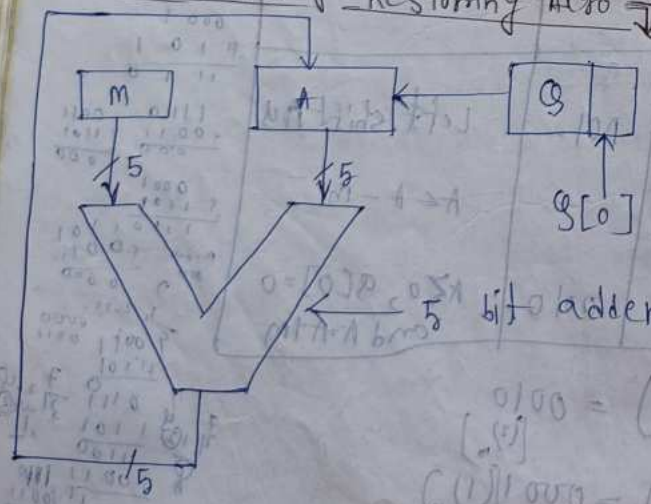
A (Remainder) = 0001 $_{(1)_{10}}$

// not needed

Non-Restoring method (Algo) = Non-restoring Division Algorithm (NRDA)

Comes from the restoring division. The restoring algorithm calculates the remainder by successively subtracting the shifted denominator from the numerator until the remainder is in the appropriate range. The operation in each step depends on the result of the previous step.

Hardware Circuit to divide two 4 bit numbers using Restoring Algo



7/2/23

Imp qns

① Digitalk, fixed point (1's, 2's, signed), Addition Subtraction using complement (10's, 9's, 1's, 2's), floating point, hardware circuits.

② Number system, conversions, add and sub using complements, fixed point representation, floating point b/ IEEE 754, hardware ckt for adders, subtractors, unsigned mul, signed mul, restoring division method. (Stallings)

Computer instruction set

Instruction set - It is a computer architecture. The design of boundary of programmers & hardware.

i) what is an instruction?
ii) what is an instruction set?

Instruction: Command that we give to CPU to perform a program.

Instruction set: the whole set of instruction.

Instruction $\propto \frac{1}{\text{program size}}$

iii) The diff of instruction & instruction set?
iv) what is instruction format?

floating point, forward checks.

system, conversions, add and

fixed point

ng division

calculate

successively

denominator

til the

appropriate

each step

of the

two 4bit

adder/subtractor

mul, mxQ → format

multiplicand → multiplier

In table, m → constant

Q → changeable

DIV, Q ÷ m → format

Q → dividend

m → divisor

In table, m → constant, Q → changeable

Prog size ∝ 1/ins. length ∝ memory ∝ execution time

ins. length ∝ 1/no. of address field

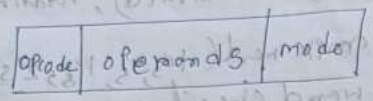
no. of ins in a line ∝ 1/execution time

to

iii) the diff of instruction & instruction set?

instruction format?

Instruction format (All-Imp)



opcode = operation code, the operation that is need to be performed

operands = on which you are going to apply your operations

Ex= ADD R1, R2, R3 (R1 = destination, R2, R3 = Source)

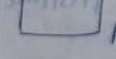
R1 ← R2 + R3 [At 1st we will keep destination always]

mode = The no. of ways by which we can specify our operation.

CPU organization

1) Single Accumulator Based CPU:

[Accumulator works as both destination & Source]



AC ← 5

ADD 5, 4

AC ← AC + 4

AC = Accumulator

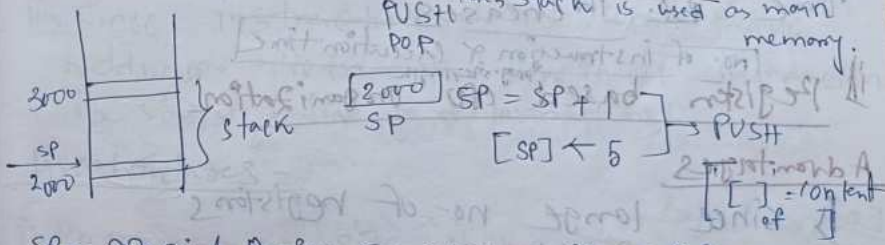
2) Register Based CPU organization =

(CPU has large no. of special general Purpose registers)

ADD R1, R2, R3

R1 ← R2 + R3

3) stack Based CPU Organization



SP = Special Purpose registers, Stack Pointer

R1 ← [SP] — [the content of that location that is held by SP]

SP = SP - 1 (POP)

[Here, CPU has a no of special Purpose registers]

Single Accumulator CPU organization

Advantages - (1) One of the operand is always held by the accumulator register. This results in shorter instruction.

(2) Uses less memory space.

(3) Instruction cycle takes less time because it saves time in instruction fetching from memory.

Disadvantages - ① when complex expression are computed, Programme size increases due to the usage of many short instructions to execute it. (programme size↑)

② As the no of instructions increases for a programme the execution time ~~is~~ increases.
(no. of instruction ∝ execution time)

ii) Register based CPU organization

Advantages

i) Since large no of registers are used in this organization, the efficiency of the CPU increases.

ii) Less memory space is used to store the programme since the instructions are used in a more compact way.

Disadvantages

① Care should be taken to avoid unnecessary ussage of registers.

② The organization involves more cost since large no of registers are used.

iii) Stack based CPU organization

Advantages

i) Efficient computation of complex arithmatic expressions.

ii) Execution of instructions is fast (execution time decreases) because operand datas are stored in consecutive memory location.

iii) Since instruction do not come have address field the length of instruction is short. (few no of instructions)

Disadvantages -

i) Programme size increases.

Instruction length

The length of an instruction depends on the no. of address field used in it.

Advantage and disadv. of using the no of address in an instruction are -

① The fewer the addresses the shorter the instruction. (no. of address field ∝ no of ins)

② Long instructions need multiple addresses usually required more complex decoding and processing circuits.

② Limiting the no. of addresses also limits the range of functions, each instruction can perform. [reduce the no. instructions (addresses) flexibility will also reduce]

③ Fewer addresses means more Primitive instructions. so longer Programmes are needed.

④ Large Programme requires longer execution time. (large memory space also)

EA Im Ind
Imd = (A+B) * (C+D)

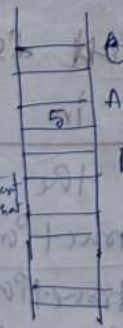
Three Address Instruction (each OPcode ~~can be addrs~~ ~~3 instruction~~) is followed by operands / addresses

ADD R1, A, B $R_1 \leftarrow M[A]+M[B]$

ADD R2, C, D $R2 \leftarrow m[C]+m[D]$

MUL X, R1, R2 $m[X] \leftarrow R_1 * R_2$

[m[ ] → Denoting memory locations]



Register Based CPU o.

Benifits ① Instruction length increases but prog. length decreases. (execution time decreases, it wants less memory space)

Advantage ① It generates shard Programmes.

Disadvantage ① It uses long instruction.

Two Address Instruction format (most popular format)

LOAD	R1, A	$R_1 \leftarrow m[A]$
ADD	R1, B	$R_1 \leftarrow R_1 + M[B]$
LOAD	R2, C	$R_2 \leftarrow m[C]$
ADD	R2, D	$R_2 \leftarrow R_2 + M[D]$
MUL	R1, R2	$R_1 \leftarrow R_1 * R_2$
STORE	X, R1	$M[X] \leftarrow R_1$

One - Address Instruction format

LOAD	A	$AC \leftarrow m[A]$
ADD	B	$AC \leftarrow AC + m[B]$
STORE	T	$m[T] \leftarrow AC$
LOAD	C	$AC \leftarrow m[C]$
ADD	D	$AC \leftarrow AC + m[D]$
STORE		

Single Accumulator Based CPU o. [No need to specify the Accumulator]

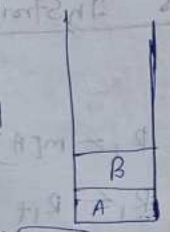
MUL T AC ← AC * M[T]
STORE X M[X] ← AC

[Here the m is X=(A+B) * (C+D) so, it does'nt make any Problem but if it is like a = (A+B) / (C+D) then it will produce Problem so, here also we have to calculate the 2nd part 1st]

Zero-Address Instruction format

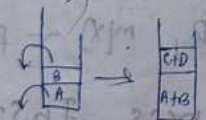
PUSH	A	TOS ← m[A]
PUSH	B	TOS ← m[B]
ADD		TOS ← A+B
PUSH	C	TOS ← m[C]
PUSH	D	TOS ← m[D]
ADD		TOS ← C+D
MUL		TOS ← (C+D) * (A+B)
POP	X	m[X] ← TOS

} Stack Based CPU o.



[TOS = Top of Stack]

[For division And Subtraction we need to do (C+D) part 1st then (A+B)]



[This is the Problem of this method, but in three & two address there is no such Problem].

① Instruction – An instruction is a set of codes that the computer processor can understand: the code usually in 1s and 0s, or machine language. It contains instructions or tasks that control the movement of bits and bytes within the processor. Example of some instruction sets – ADD – Add two no. together.

② Instruction set – It is the set of instructions that a specific CPU actually supports. Instruction set architecture (ISA) is the set of instructions that a CPU (that implements that ISA) might or might not support.

③ Difference between instruction & instruction set – An instruction is a command that we give to CPU to perform a programme. Instruction set is a set of instructions, again, that a specific CPU actually supports.

⑤ Instruction length – It tells the total no. of instructions set defined in the processor.

① Instruction format: In computer architecture, the instruction format is defined as standard machine instruction format that can be directly decoded and executed by the CPU. This instruction format is simply a sequence of bits (binary 0 or 1) contained in a machine instruction that defines the layout of the instruction.

10/2/23

$X = (A+B*C)/(D-E)$

① Three Address:

MUL R1, B, C $R_1 \leftarrow m[B] * m[C]$

ADD R2, A, R1 $R_2 \leftarrow m[A] + R_1$

SUB R3, D, E $R_3 \leftarrow m[D] - m[E]$

DIV R2, R3 $R_2 \leftarrow R_2 / R_3$

② Two Address:

LOAD R1, B $R_1 \leftarrow m[B]$

MUL R1, C $R_1 \leftarrow R_1 * m[C]$

ADD R1, A $R_1 \leftarrow R_1 + m[A]$

LOAD R2, D $R_2 \leftarrow m[D]$

SUB R2, E $R_2 \leftarrow R_2 - m[E]$

DIV R1, R2 $R_1 \leftarrow R_1 / R_2$

STORE X, R1 $m[X] \leftarrow R_1$

③ One-Address:

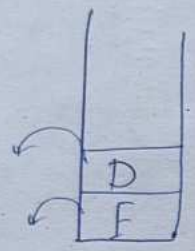
LOAD	B	AC ← m[B]
MUL	C	AC ← AC * m[C]
ADD	A	AC ← AC + m[A]
STORE	T	m[T] ← AC
LOAD	D	AC ← m[D]
SUB	E	AC ← AC − m[E]
DIV	T	AC ← AC / m[T]
STORE	X	m[X] ← AC

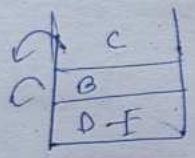
④ Zero-Address:

PUSH B	TOS ← m[B]
PUSH D	TOS ← m[D]
SUB	TOS ← D − E
PUSH B	TOS ← m[B]
PUSH C	TOS ← m[C]
MUL	B = C * B
ADD	A + B * C
DIV	(A + B * C) / D − E
POP X	m[X] ← TOS

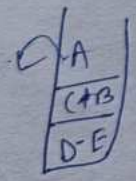
PUSH A TOS ← m[A]

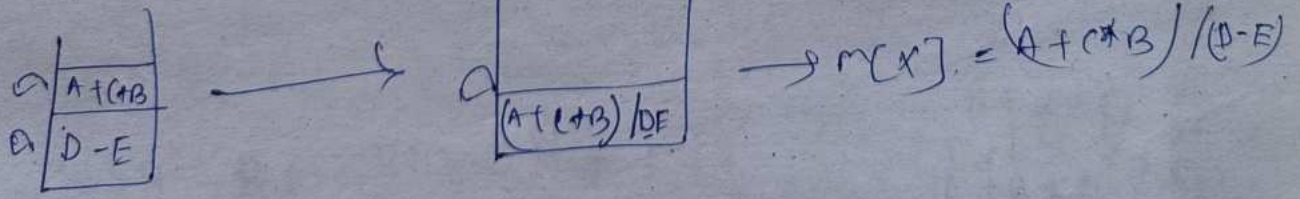
(2)


D
F


C
B
D-F


C+B
D-F


A
C+B
D-E


A+C+B
D-E
(A+C+B)/DE
m[X] = (A+C*B)/(D-E)

Addressing modes (Adds Imp)

Defination- the different ways in which the location of an operand is specified in an instruction are referred to as an addressing modes.

Advantages of using diff. Addressing modes

① It gives Programmer the flexibility and the versitility with respect to the no. of instructions & execution time by providing various addressing modes.

② To reduce the length of the instruction or the size of the Programme.

Class-I:- No addressing field is specified as a part of instruction

1) Implied Addressing mode:-

In this Addr. mode the operandes are indicated implicitly by the instruction this mode is very popular in 8 bits microprocessors like intel 8085.

Ex- 1) CMA → (Complement Accumulator) → will access Accumulator, as nothing is given that on what it have has to be done.

2) RAL → Rotate left

Diagram

OPcode

no operands → no OPCode

Advantage- takes very little memory space

Disadvantage- No flexibility, we can not do more operations

2) Stack Addressing mode

stack organized computers use stack addressing modes. In this addressing mode all the operands for an instruction are taken from the top of stack.

Ex- 1) ADD
2) SUB

3) Immediate Addressing mode

In this mode the operand is mentioned explicitly in the instruction. An immidiate mode instruction contains an operand value rather than an address of it in

the address field.

Ex – 1) ADI 05H (operand)
(Add Immidiate) to the Accumulator

AC ← AC + 05

[ADI = Instruction for Intel 8085]

[If nothing is said then the by default operand is AC]

[05H = 05 Hex]

2)

MVI B, 06H
(Move Immidiate)

B ← 06H

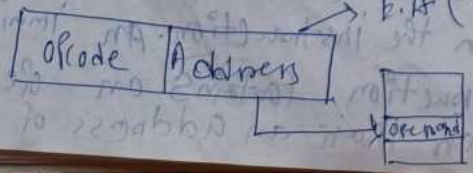
[B = Register]

Addressing mode 13/2/23

Class-II : Address field is a memory address:

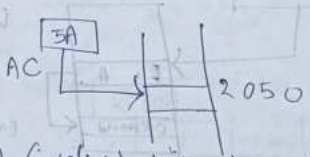
1) Direct or Absolute Addressing mode:

In this mode the instruction contains the memory address of the operand explicitely. Thus the address part of the instruction is called the direct address.



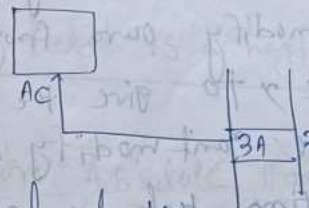
Ex

1) STA 2050H
(Store Accumulator Direct)



the 5A (value) will be stored at the to memory location 2050.

2) LDA 2050H
(Load Accumulator Direct)

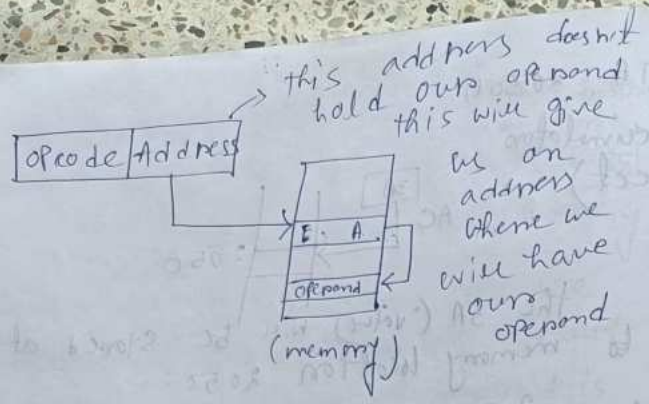


the value present at the memory location 2050 will be load at AC.

Effective address – The location where the operand is stored or load is called effective address.

2) Indirect Addressing mode

In this mode the instruction gives a memory address in its address field which holds the address of the operand. Thus the address field of the instruction gives the address where the effective address is stored in memory.



Why this Process / why we need 2 addresses?

1) To modify our prog. during run time, to give the flexibility. As we can't modify our prog during run time but we can change the address that is very useful for building pointers also.

2) It gives the Programming flexibility and this is very useful in C for building Pointers, useful for cleaning Pointers in C Programming.

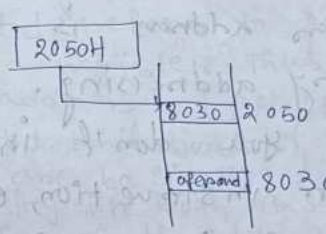
3) In this mode there is a scope of changing the address during run-time without changing the instruction content.

3) this organisation involves more space.

EX-

1) STAX B

Store Accumulator Indirect)



Im Rm

CLASS-III - Address field is a register address

1) Register Direct and Absolute Addressing mode:

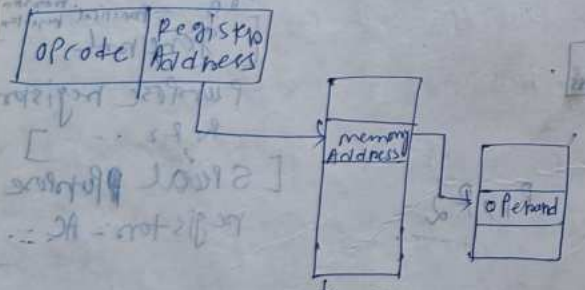
In this mode the processors registers holds the operands. The address field is a register field which contains the operands required for the instruction. This mode is useful to a long programme to storing the intermediate results in the registers rather than in memory. This will result in fast

execution since register accessing is faster than memory accessing.

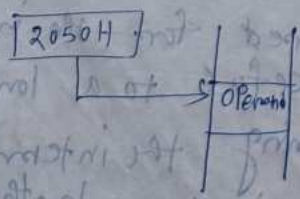
Register address is faster than memory addressing.

— As you don't increase (size) the larger instruction, extra cycles, more of the cache line burned for instruction.

2) Register indirect Addressing mode —



1) STAX, B



Just as register addressing is analogous to direct addressing, register indirect addressing is analogous to indirect addressing. In both cases, the only difference is whether the address field refers to a memory location or a register. Thus, for register indirect address. [EA = (R)]

The advantages and limitations of register indirect addressing are basically the same as for indirect addressing. In both cases, the address space limitation of the address field*

Class IV — Address field doesn't contain effective address. E.A = Address part of Instruction + Content of special CPU register

1) Relative Addressing mode or PC-relative Addressing mode —

[PC = Program Counter]

Opcode	offset

E.A = Content of PC + offset
= 2050 + 100

This addressing mode is useful for branching instruction (condition or unconditional branching) & jump instructions (goto)

In this mode the effective address is obtain by adding the content of the PC with the address part of the instruction. The instruction specifies the effective address as a relative position of the current instruction address.

Ex

1) JZ 20 → PC = PC + 20
(jump on zero) 20 is the effective address. This is address part of the instruction.

This mode is used for specifying branch address in a branch instruction.

* is overcome by having that field refer to a word length location containing an address. In addition, register indirect addressing uses one less memory reference than indirect addressing.

Class-X

17.2.23

2) Indexed Addressing mode:-

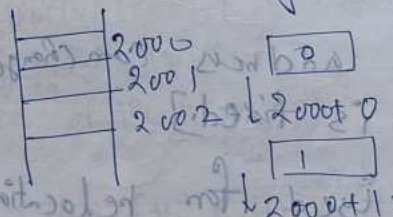
opcode	offset

[ ] Index register

E.A = ~~[PC]~~ + offset + [IR]

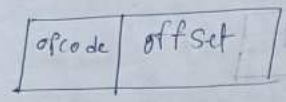
In this mode the effective address is determind by adding the content of the index register in the address part of the instruction. This mode is useful in accessing an array operand.
[Here we can change the index, but can't change B.A]

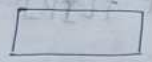
opcode | offset → Providing the base address of the array → 2000



Advantage - we can get the flexibility that we can chang the index of the array.

3) Base Addressing mode:-

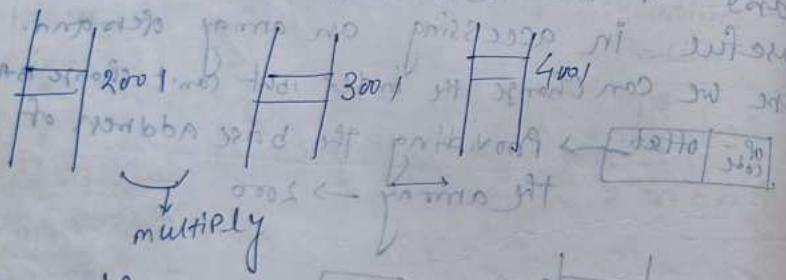




Base Register

EA = [Base Register] + offset

[Base register provides the Base Address of an array.]

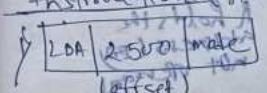


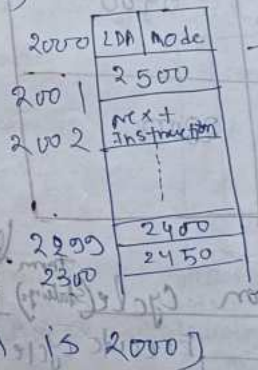
[Here the base address can change but the index is fixed.]

This mode is used for relocation of the programmes in the memory. Here the content of the B.R holds the starting address of a memory array of operands and the address part gives a displacement. The Base Addressing mode has an advantage over indexed addressing mode with respect to the size of the instructions because size of instructions in the 1st case is smaller than the size of instruction in the 2nd case.

Example -

1) Suppose the 2 word instruction stone at address 2001 is LDA 2500





this address will be given by ins we have to draw the diagram by following instruction

2) PC = 2000
(As the B.A is 2000)
(Address of the PC = 2000)

3) R1 = 2300 [the address of Register2 E.A]

4) XR = 20 [Index Register]

If register indirect the the addr. of register holds E.A, so Addr. of register = E.A]

[I.R.A. tro. = E.A = 2500 + 20 = 2520]

[For 2 word instruction = [PC] = B.A + 2 = 2000 + 2 = 2002]

← this will be given that what we have to calculate

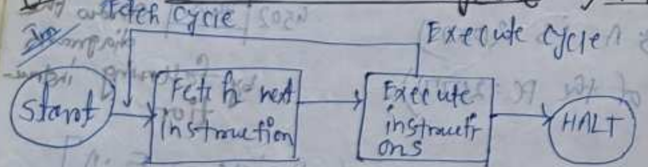
Add mode	E.A	Content of AC
Immediate	2001	2500
Register	–	2300
~~Indirect~~ Register Indirect	2300	2450
Direct	2500	2700
Indirect	2500	2700
Indirect	2700	2250
Relative	[PC] + 2500 = 2002 + 2500 = 4502	3400
Indexed	3500 + 20 offset [20] = 2720	2800

Content of AC can be said operand as the E.A holds the operand.

As nothing is said by default we have to take AC, AC is a g.p.r that holds address and address (E.A) holds operand ∴ Content of AC = Operand

→ In address part, the register is present & R holds the address (EA) not operand

Instruction Cycle (Stallings) 10/3/23 (from All-imp)



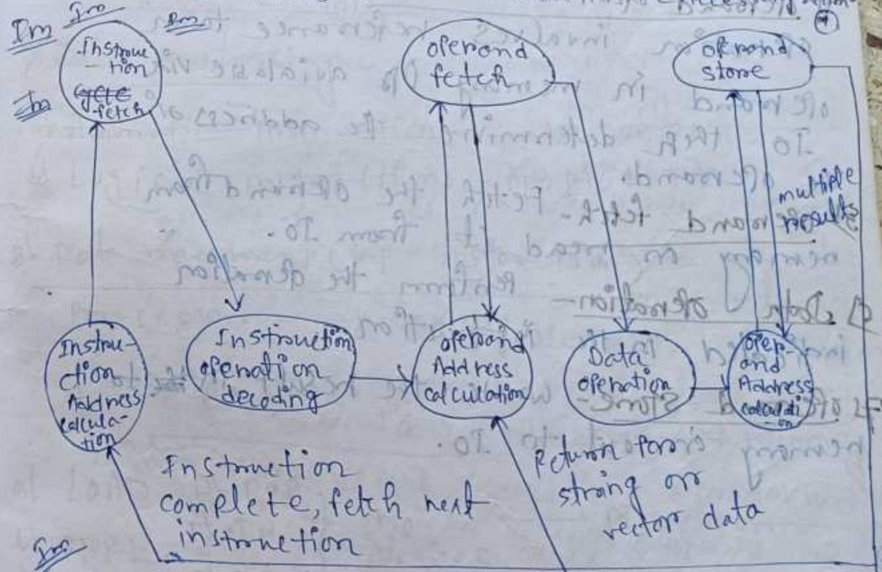
Basic Instruction cycle

~~The time~~ executing a single execution is the time taken for called instruction cycle. The processing performed for a single execution is called instruction cycle. — Instruction cycle.

It consist of 2 steps – 1) fetch cycle (fetch instr. from memory and store it in instruction register) → (pending) 2) Execution cycle (whenever we are getting some data it stored in MAR (it store the address of the operand), the operand is fetched by MAR) and then the execution happens)

1) Fetch cycle –
The processor fetchs instructions from memory one at a time and stores it into a register of the processor known as instruction register.

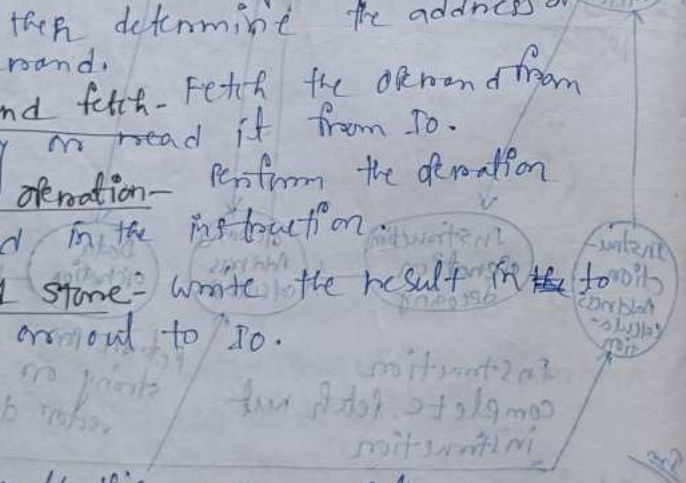
2) Execute cycle – the instruction contains bits to specify the action, the processor is to take. The processor interprets the ins. and performs the necessary action.



Instruction cycle state Diagram (details of Basic instruction cycle)

Operand address calculation is done by addressing mode.
Instruction address is given by PC.
vector data → Array

1) Instruction address calculation – determine the address of the next instruction to be executed.
2) Instruction fetch – Read instruction from it's memory location into the processor.
3) Instruction operation decoding – Analyse instruction to determine type of operation to be performed and operands to be used.
4) Operand address calculation – If the operation involves reference to an operand in memory or available via I/O then determine the address of the operand.
5) Operand fetch – Fetch the operand from memory or read it from I/O.
6) Data operation – Perform the operation indicated in the instruction.
7) Operand store – Write the result in memory or out to I/O.



In general this actions falls in 4 categories –

1) Processor – memory – Data may be transfed from processor to memory or vice versa.
2) Processor – I/O – Data may be transfer to or from a peripheral device by transfering between the processor and an I/O module
3) Data – Processing – The processor may perform some arithmetic or logic operation on data.
4) Control – An instruction have to specify that the sequence of execution be altered. (Branching)

Instruction Types

1) Data processing –
   a) Arithmatic instructions – ADD, SUBTRACT
   b) Logical instructions – AND, OR, NOT
2) Data movement (imp instructions)
   Processor ⟶ memory
   Processor ⟵ memory
   Processor ⟶ I/O
   I/O ⟶ Processor
   Processor ⟷ memory
   a) LOAD d) POP
   b) STORE e) IN
   c) PUSH f) OUT
   I/O devices ⟷ Processor
3) Control
   change the sequence of the instruction
   a) JUMP (conditional)
   b) JUMP (Unconditional)
   c) NOP (no operation) → 4 CLK operation (Clock (CLK) → 111)

d) WAIT
e) HOLD
f) HALT

[DMA Processor = Direct Memory Access]

← (Definations & examples of all)

Instruction types

1) Data Processing - It means to process the data, ie to convert its format. As we all know data is very useful and often it is well presented and it becomes informative & useful. Data processing system is also referred as information system.

Data → Process → Information

In simple word data processing can be expressed as: 1) process of conversion of data in the computer understandable format. 2) The sorting or processing of data by a computer.

Advantages - 1) Highly efficient 2) Time-saving 3) High speed, 4) Reduces errors.

Disadvantages - 1) Large power consumption, 2) occupies large memory, 3) the cost of installation is high, 4) wastage of memory.

a) Arithmetic instructions - Arithmetic instructions are the instructions which perform basic arithmetic operations such as addition, subtraction & a few more.

i) ADD - The content of operand are added to the content of the accumulation and the result is stored in Ac.

ii) SUB - Any 8bit data or the contents of a register or contents of a memory location can be subtracted from the contents of the Ac.

b) Logical instructions - They are the instructions that perform basic logical operations such as AND, OR etc. In the 8085 microprocessor the destination operand is always the Ac. The logical operation works on a bitwise level.

2) Data movement (imp) - It is the ability to move data from one place in your organization to another through technologies that include extract, transformation, load (ETL), extract, load, transform (ELT), data replication and change data capture (CDC) primarily for the purpose of data migration and data warehousing.

EX - The transfer of vector operand in and out of vector register is an example of data movement.

Function - MOV (move) transfers a byte, word or doubleword from the source operand to the destination operand.

Importance - Data movement, in all of its forms, is an enabling technology rather than a solution in its own right: it is used to populate data warehouses and to exchange information with business partners and between applications, it is used to provide high availability,

i) LOAD = A load operation caries data from main memory into a register.

ii) STORE - Stores are very important in carrying out day-to-day operations. The objective behind Stores is the continuous supply and production of goods and services.

iii) PUSH - The push operation is used to insert a new piece because the stack is not yet full. The SP (stack pointer) is raised by one, and the location of the next highest word is stored in the stack pointer.

iv) POP - It delete an element from the stack.

v) OUT - TO show the o/p ~~In~~ to show the i/p

3) Control - It understands commands and instructions. It regulates the flow of data within the processor, change the sequence of the instruction.

a) Jump (conditional) - Branching in Assembly. It does a goto somewhere if the two values satisfy the right condition (Trans. address only if the condition is statisfied)

b) Jump (unconditional) - Transfers the program sequence to the described memory address.

c) NOP (NO operation) - The NOP instruction does nothing. Execution continues with the next instruction. It used to generate a delay in execution or to reserve space in code

memory.

d) WAIT - It will suspend execution of the calling thread until status information for one of its terminated. A wait state is a delay experienced by a computer processor when accessing external memory or another device that is slow to respond.

e) HOLD - It is used to quickly accept, store, and transfer data and instructions that are being used immediately by the CPU.

f) HALT - It puts the 80386 in a HALT state by stopping instruction execution.

It may be used to support data preparation (Imp – mono)

RISC vs CISC

Reduced instruction set Computers

Complex instruction set computers

- difference between RICS & CICS architecture
- characteristics of RICS & CICS architecture

RICS	CICS
1) multiple register sets, often consisting of more than 256 registers.	1) single register set typically 6 to 16 registers total.
2) Three register operands allowed per instruction (e.g. ADD R_1, R_2, R_3)	2) one or two register operands allowed per instruction (e.g. ADD R_1, R_2)
3) Hardwired control	3) multiple cycle instructions ~~Programmed control~~
4) Highly pipelined	4) less pipelined
5) single-cycle instructions (except for load and store)	5) multiple-cycle instructions
6) simple instructions that are few in no.	6) many complex instructions
7) fixed length instructions.	7) variable length instructions
8) Complexity in compiler.	8) Complexity in microcode
9) Few addressing modes.	9) many addressing modes.
10) only load and store instructions can access memory	10) many instructions can access memory.

Characteristics of RISC –

1) simpler instruction, hence simple instruction decoding.
2) Instruction comes under size of one word.
3) Fewer Data types
4) Simple Addressing modes
5) more general-purpose registers.
6) A pipeline can be achieved
7) Instruction takes a single clock cycle to get executed.

Characteristics of CISC –

1) Complex instruction, hence complex instruction decoding.
2) Instructions are larger than one-word size.
3) less no. of general-purpose registers as operations get performed in memory itself.
4) Complex Addressing modes
5) more Data types.
6) Instruction may take more than a single clock cycle to get executed.

- RISC – Reduce the cycles per instruction at the cost of the number of instructions per program.
- CISC – the CISC approach attempts to minimize the no. of instructions per Prog. but at the ~~less~~ cost

3) Fixed
4) simple Addressing
5) more general-purpose registers.
...line can be achieved
... a single clock

of an increase in the no of cycles per instruction.

$$\text{CPU time} = \frac{\text{seconds}}{\text{program}} = \frac{\text{Instructions}}{\text{program}} \times \frac{\text{cycles}}{\text{Instructions}} \times \frac{\text{seconds}}{\text{cycle}}$$

Set of orthogonal instructions make ✓notes

Orthogonal instruction set (mano & stallings)

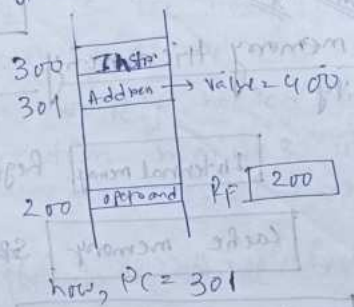
An orthogonal instruction set is an instruction where all instruction can use all instruction addressing modes. [It is an instruction set architecture where all instruction types can use all addressing modes. It is "orthogonal" in the sense that the instruction type & the addressing mode vary independently]

24/3/23

Ex-0

An instruction is stored at loc 300 with its address field at loc 301. the address field have same as the value 400. A processor register R1 contains the no. 200. Evaluate the effective address if the addressing mode of the instruction is- i) direct
ii) Immidiate
iii) Relative
iv) RI (Register Indirect)
v) Indexed with R1 as the index R.

given,



now, PC = 301

i) Direct addressing mode

EA = 301

ii) Immidiate = 400 = E.A → As address field holds operands rather than address so there is no E.A or E.A = operand

iii) Relative = PC + R1 = 301 + 200
= 501 = E.A

iv) Register indirect = 200 = E.A → [As the address part holds some R that holds E.A]

v) Indexed with R1 as index R = R1 + the content of the instruction
= 200 + 400
= 600 = E.A

[he will not take the address as we usually take the operand mentioned at address part]

memory organisation (mano & stalings)

RAM design, cache memory

External & internal memory

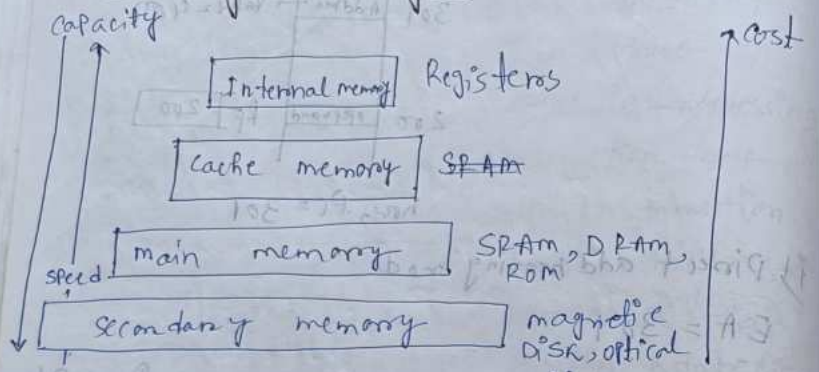
memory = opcode memory of the RAM

When we are seeing memory we will consider some things- i) Capacity of the memory
ii) Speed (Time the memory speed
iii) Bandwidth is (Given to internal memory)

CPU utilization should be maximum.

(manno ch-12)
(00) Stallings

(1) memory Hierarchy :-



The memory heirarchy system, consist of all storage devices used in a computer system and is broadly divided into 4 groups - i) Secondary (Auxiliary) memory

- The slow speed & low cost device that provides backup storage are called secondary memory, when a prog. not residing in main memory is needed to execute, it is transferred from secondary memory to main memory. Programs not currently needed in main memory are transferred into secondary memory. S.M is 100 times slower than from CPU. ex = OD, MD.

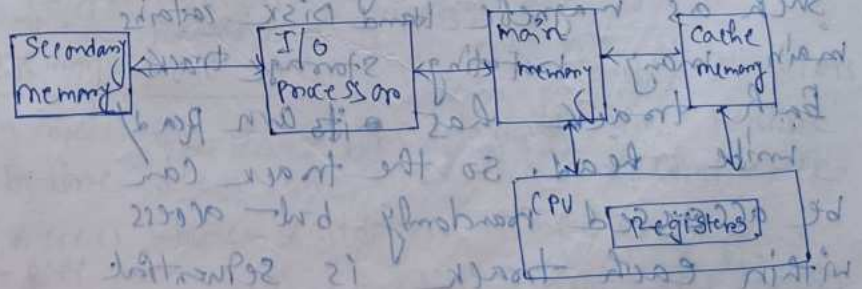
memory = main/primary memory

ii) main memory - This is the memory that communicates directly with the CPU. only progs and data currently needed by the CPU, for execution resides in the memory.

iii) Cache memory - This is a special high speed main memory, that stores copies of the prog. that are frequently needed by the CPU. (Ex- looping in prog.).

iv) Internal memory - This memory refers to the high speed registers used inside the CPU. This registers holds temporary results when a computation is in process. (no speed mismatched) (same as CPU)

Interconnection b/w memory & CPU



Block diagram

SM is not online with CPU. If we want to put something in main memory we have to go through by I/O memory.

Access method ← (mano & stallings)

i) Sequential Access — In this method the memory is accessed in a specific linear sequential manner.
Ex- MD, OD, Tape, optical memory, magnetic tape

ii) Random Access — In this mode of access any location of the memory can be accessed randomly. Ex - Semiconductor memory like RAM, ROM are of this type.

iii) Direct Access — This method is a combination of previous two Access methods. memory devices such as magnetic Hard Disk contains many rotating storage tracks. Each track has its own Read/write head. So the track can be accessed randomly but access within each track is sequential

iv) Associative Access — (cache memory)
This is a special type of Random Access method that enables one to make a comparison of the desired bit location within a word for a specific match & to do this for all words simultaneously. Cache memory uses this type of access mode.

Addressing modes End

Access time (Latency) - For RAM (Random Access memory) this is the time it takes to perform a read/write operation, that is the time from the instant an address is presented to the memory to the instant that data have been stored or made available for use. (From the time the address is given to the time the data is available - Latency)

From the time the address is given to the time the data is available is 50 ns (nano seconds)

Memory cycle time - This is consist of access time & any additional time required before a 2nd access can commence. (Access time, addition time)

- Defi

Access time - It is the time delay or latency between a request to an electronic system and the access being completed or

(Bandwidth)

iii) Transfer rate - This is the rate at which data can be transferred in and out of a memory units for RAM. This is expressed as the no. of bytes or words per second. (mBBS/second) ③

the requested data returned.

27/3/23 (manno)

General Purpose Computer

main memory

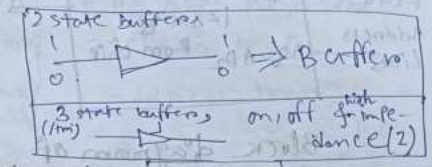
(Central storage device = main memory

- to RAM (Read (volatile) write memory)
  - Compiler, loader, linker:
    - Stored the Prog to specific position of memory & give the starting address
    - Power on self test = The sound while opening the PC (Personal computer)
- Rom (Read only memory) (not volatile) (Bootstrap loader)
  - It is the 1st prog that will run while we will open our computer

Rom can not be modified, OS is loaded in Rom. Control memory of a basic general P. RAM.

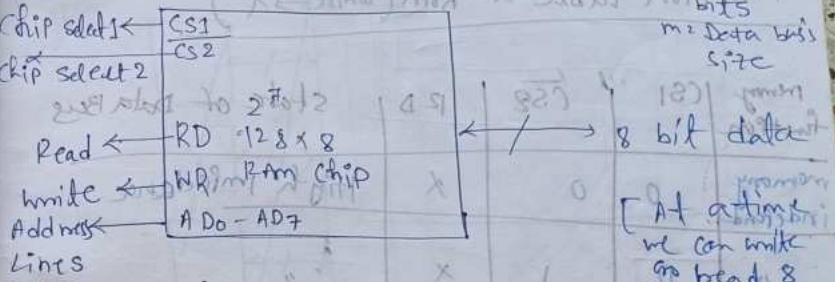
Control memory of the Special Computers = Rom.

Ram → SRAM
Ram → DRAM



Basic RAM chips (manno)

$2^n \times m$ → n = no. of bits, m = Data bus size



[At a time we can write or read 8 bit data]

Block diagram of RAM chip [m = 8]

Memory function	CS1	CS2	RD	WR	State of Data Bus
memory independent	0	0	X	X	High Impedance
"	0	1	X	X	"
Output data from RAM	1	0	0	X	Read
I/P data to RAM	1	0	0	1	write
memory Inhibit	1	0	X	X	High impedance
memory inhibit	1	1	X	X	High impedance

Function table of RAM

Bootstrap loader = A bootloader, also spelled as boot loader or called boot managers and bootstrap loader, is a computer program that is responsible for booting a computer. when a computer is turned off, its software — including operating systems, application code, and data — remains stored on non-volatile memory.

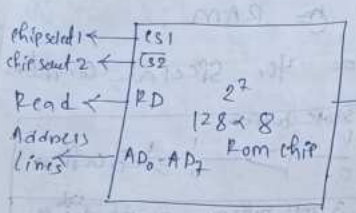
Complier & interpreter - A compilers takes a program as a whole. An interpreter takes single lines of a code. Output - the compiler generate intermediate machine codes. the interpreters never generate any intermediate machine codes.

loader - In computer systems a loader is the part of an operating system that is responsible for loading programs & libraries. It is one of the essential stages in the process of starting a program, as it places programs into memory & prepares them for execution.

linker - In computing, a linker or link editor is a computer system program that takes one or more object files (generated by a compiler or an assembler) and

function table of

Combines them into a single executable file, library file or another "object" file.



Block diagram of Rom

Function table of Rom (write → ✗)

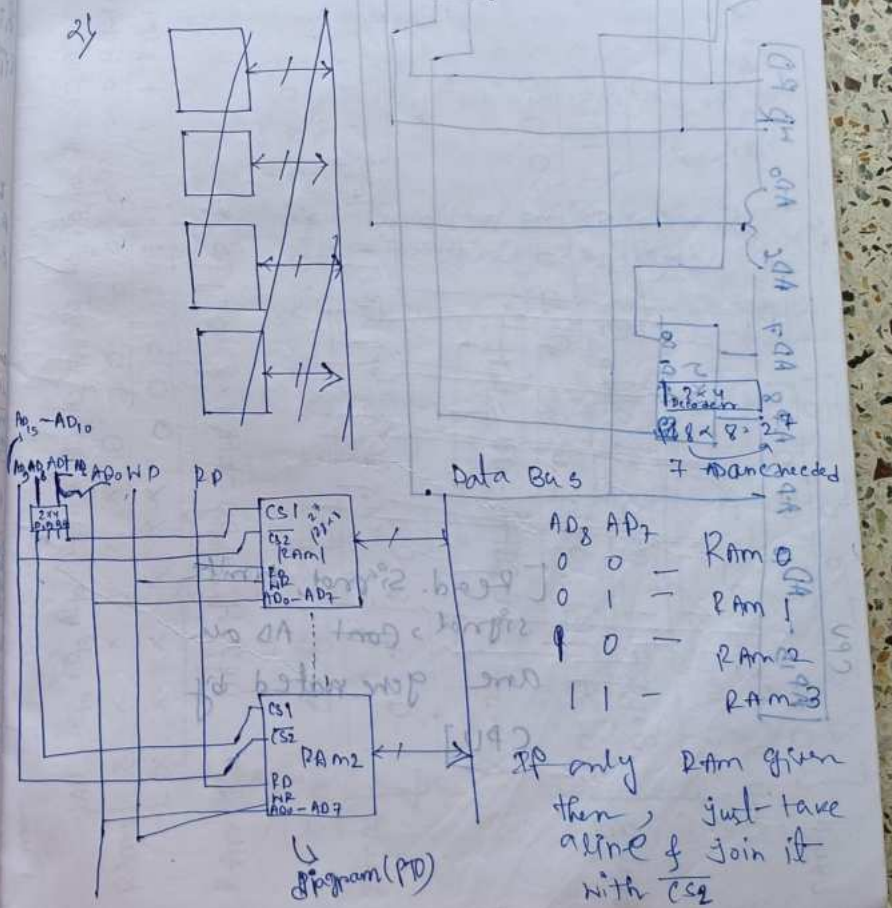
memory function	CS1	CS2	RD	state of Data Bus
memory independ	0	0	x	High impedance
	0	1	x	"
output data from rom	1	0	1	Read
~~s/p data~~ to memory inhibit	1	0	0	High impedance
memory inhibit	1	0	x	High impedance
"	1	1	x	"

① Suppose the computer system needs 512 bytes of RAM each RAM chip is 128 × 8

1) How many RAM chips are required?
2) Draw the memory address map.
3) Give the memory address range. (Imp)

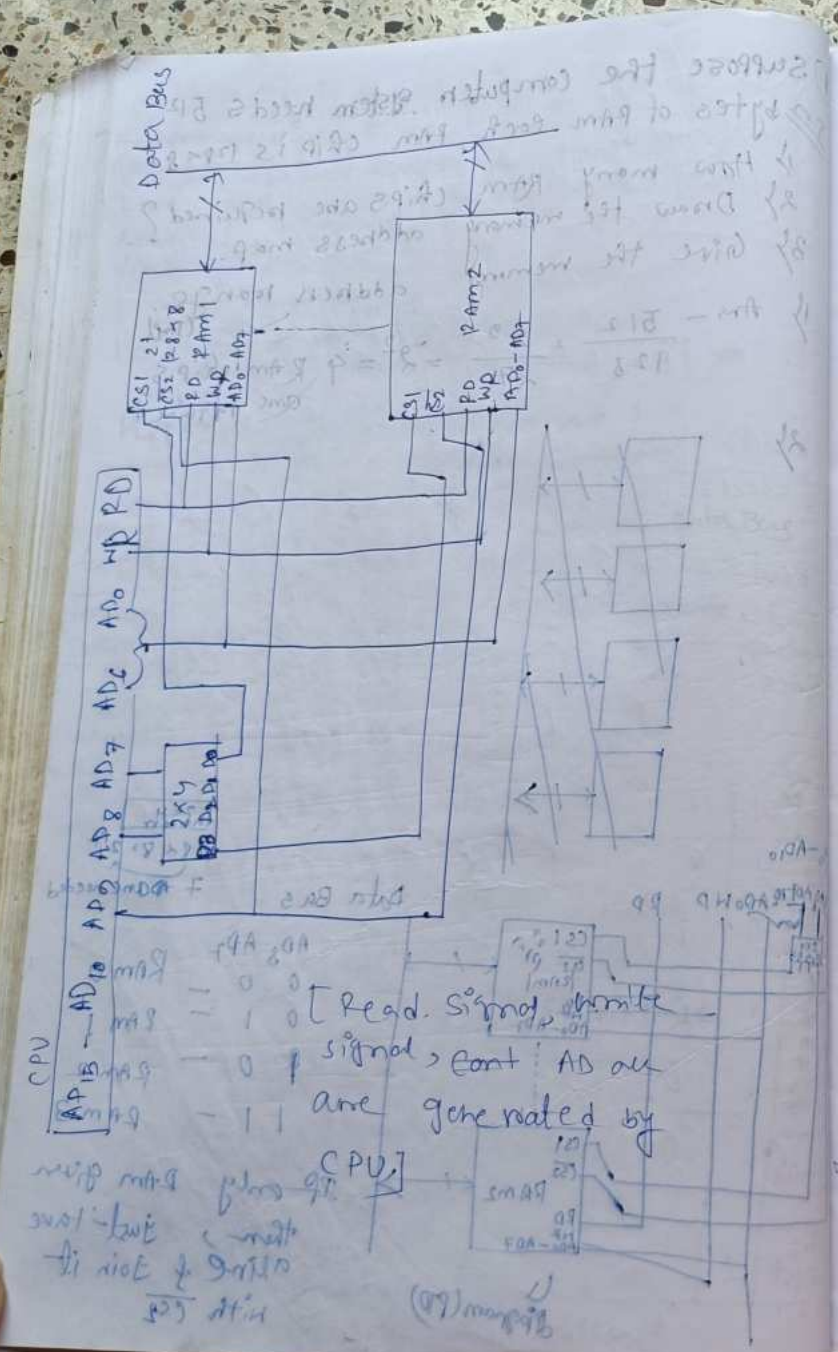
1) Ans – $\frac{512}{128} = \frac{2^9}{2^7} = 2^2 = 4$ RAM chips are required.

2)



AD8	AD7	
0	0	– RAM 0
0	1	– RAM 1
1	0	– RAM 2
1	1	– RAM 3

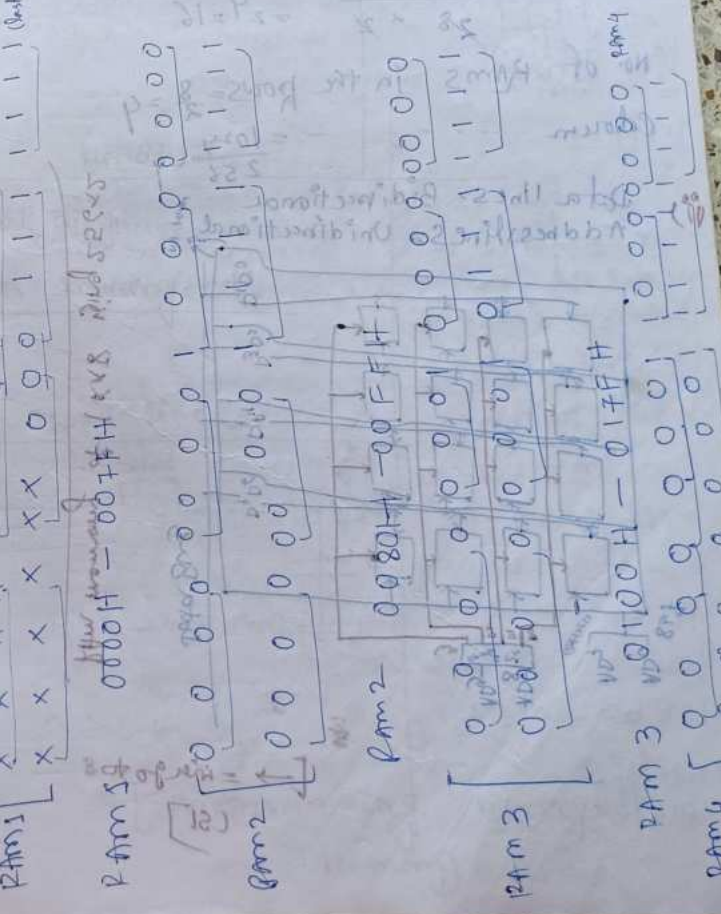
If only RAM given then, just take a line & join it with $\overline{CS_2}$



31/3/23

(is from homogenious case)

RAM1	0000H – 007FH
RAM2	0080H – 00FFH
RAM3	0100H – 017FH
RAM4	0180H – 01FFH



Hetrogenious / cafe

Type - II (Homogenious)

① Construct a RAM-type memory of size 1K×8 using RAM chip of size 256×2.

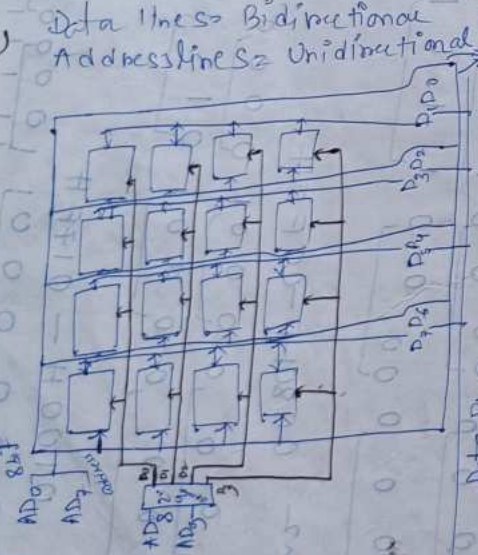
i) How many RAM chips are required?

$$\frac{1K \times 8}{256 \times 2} = \frac{2^{10} \times 8}{2^8 \times 2} = 2^4 = 16$$

No. of RAMs in the rows $= \frac{8}{2} = 4$

Column $= \frac{1024}{256} = 4$

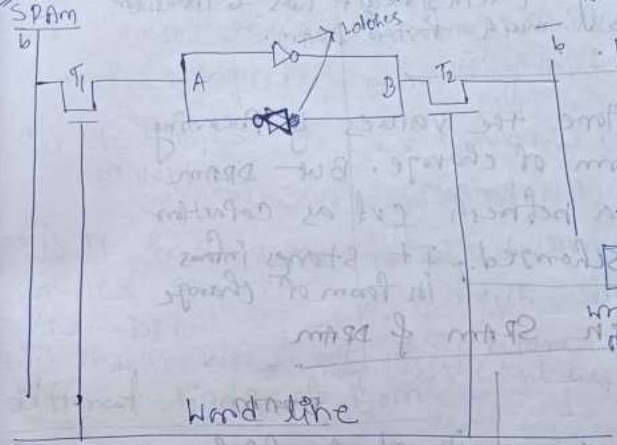
ii) Data lines = Bidirectional
Address lines = Unidirectional



[→ = will go to: CSI]

SRAM & DRAM (Stallings)

SRAM memory cell — As long as the power is applied it will work.



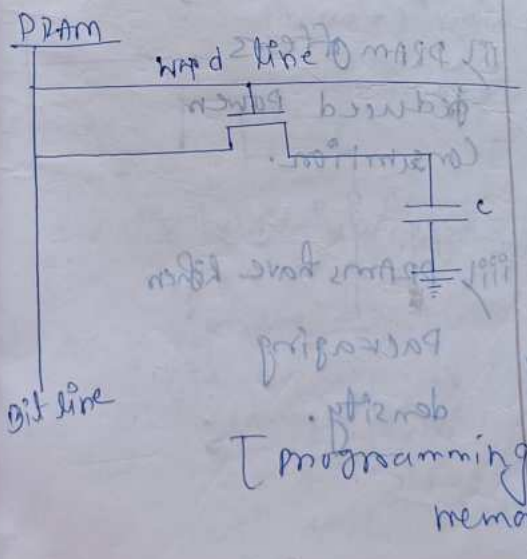
[b, $\bar{b}$ = complement value of bit]

[T_1, T_2 = Transistor]

[Latch can written 1 bit of instruction]

① block diagram

DRAM memory cell



[programming memory = Rom memory]

SRAM Advantages & Disadvantages

Low Power Consumption. But it is very expensive as it has 6-transistor. SRAM is much faster DRAM.

DRAM

We store the values by presenting in the form of charge. But DRAM needs a refresh ckt as capacitors slowly discharged. It stores infoms in form of charge

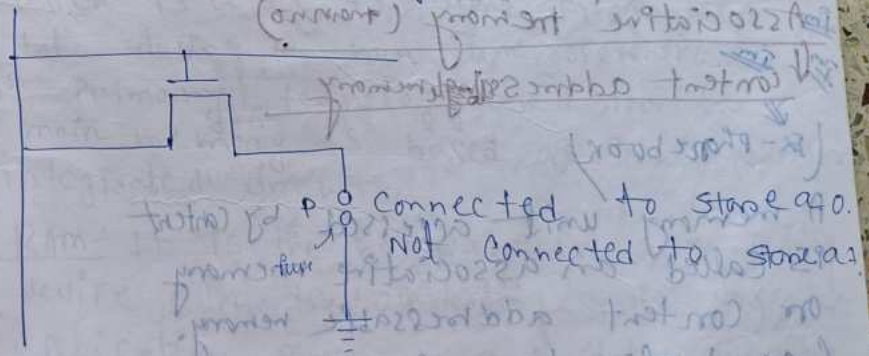
Diff. between SRAM & DRAM

SRAM	DRAM
i) It has lowers access time which means it is faster compare to DRAM.	i) It has higher access time comp are to SRAM.
ii) It requires constant Power supply so it consumes more Power.	ii) DRAM offers reduced Power Consumption.
iii) SRAMs have lower Packaging density.	iii) DRAMs have higher Packaging density.
iv) It is costly of than DRAM.	iv) It is less costly than SRAM.
v) It doesn't require a refreshing ckt.	v) It requires a refreshing ckt.
vi) It stores bit as voltage	vi) It stores bit as charge.
vii) It has a complex structure than DRAM.	vii) It's structure is much simpler than as compared to SRAM.
viii) They are used in cache memory	viii) They are used in main memory.

Different kinds of Rom

i) PRom (Programable Rom)

PRom cell - It contain a fuse that only reprogramable by once.



(ii) EPROM (Erasable Programmable Rom)
take it out

we have to take out the chip under the UV Rays to delete the whole chip. But this is disadvantage.

(iii) EEPROM (Electrically Erasable PRom)

A special version of EEPROM = Flash memory, Expensive. very high Packaging memory. If we get it out from system the info will not be deleted, it will remain stored

(iv) Flash memory

Prom working principle (fuse). After that it is a EEPROM.

Associative memory (memo)

Content addressable memory

(Ex- Phone book)

A memory unit accessed by content is called an associative memory or Content addressable memory. (CAM). This type of memory is accessed simultaniously & in parallel on the basis of the data content rather than by specific address or location. When a word is written in associative memory no address is given. The memory is capable of finding an empty unused location to store the word. When a word is to be read from an associative memory, the content of the word or the part of the word is specified. the memory locates all words which match the specified content and marx them for reading.

~~Virtual memory~~

Main memory - It acts as the central Storage unit in a Computer system. It is a relatively large and fast memory which is used to store programs & data during the run time operations. The primary technology used for the main memory is based on Semiconductor integrated circuits.

RAM - It is the hardware in a Computing device where the operating system (OS), application programs and data in current use are kept so that they can be quickly reached by the device's processor. RAM is the main memory in a computer.

ROM= The memory from which we can only read but cannot write on it. This type of memory is non-volatile. The information is stored permanently in such memories during manufacture.

SRAM = (Static RAM) is a type of RAM that retains data bits in its memory as long as power is being supplied. Unlike dynamic RAM, which must be continuously refreshed, SRAM does not have this requirement, resulting in better performance and lower power usage.

DRAM - It stands for dynamic random access memory, and it's a specific type of RAM. All computers have RAM, and DRAM is one kind of RAM we see in modern desktops and laptops

Advantage & disadvantages of SRAM

Advantages - i) True digital behaviour. Same as logic gates.
ii) High speed; similar to speed of single latch.
iii) Easy access circuitry; decoders, plus mux/demux to select bits from word if word too longe.

Disadvantages
i) High power consumption; same as latch.
ii) High heat generation; direct consequence of high power consumption.
iii) High cost; a chip full of SRAM requires an expensive package to remove all the heat; costly.

Advantages & disadvantages of DRAM:

Advantages - i) high density (capacity)
ii) cheaper cost per bit, and
iii) lower power consumption per bit

Disadvantages - i) It must be ~~the~~ refreshed periodically, due to the fact that capacitor cell loses its charge.
ii) while it is being refreshed the data can't be accessed

Features of ROM -

i) It is non-volatile memory
ii) Information stored in ROM is permanent.
iii) Information and programs stored on it, we can only read.

iv) Infos & progs are stored on Rom in binary format.
v) It is used in the start-up process of the computer.

Advantages of Rom

i) It is cheaper than RAM and it is non-volatile memory.
ii) It is more reliable as compared to RAM.
iii) Its circuit is simple as compared to RAM.
iv) It doesn't need refreshing operation time because it is static.
v) It is easy to test.

Disadvantages of Rom

i) It is a read-only memory, so it cannot be modified.
ii) It is slower as compared to RAM.

Features of RAM

i) It is volatile in nature.
ii) It is known as the Primary memory of the Computer.
iii) It is expensive as the memory can be accessed directly.
iv) It is the fastest memory, therefore it is an internal memory for the computer.
v) The speed of computer depends on RAM.

Advantages of RAM

~~An advantage of RAM is that~~
i) operating speed is faster
ii) mos memories are more economical than magnetic core for small systems.
iii) power dissipation is very low.
iv) Read out of RAM doesn't affect the contents stored.

Disadvantages of RAM

A disadvantage is that it is volatile, information can be lost when it is not saved.

MRom (masked RAM) – we know that Rom is as old as semiconductors technology. MRom was the very 1st Rom that consists of a grid of word lines and bit lines joined together by transistors switches. this type of Rom data is physically encoded in the IC and only be programmed during fabrication. It

was not so expensive.

PROM – PROM is a form of digital memory. In this type of ROM, each bit is locked by a fuse or anti-fuse. The data stored in it are permanently stored and can't be changed or erasable. It is used in low-level programs such as firmware or microcode.

EPROM – It is also called EROM, is a type of PROM but it can be reprogrammed. The data stored in EPROM can be erased and reprogrammed (if it is limited). Before the era of EEPROM and flash memory, EPROM was used in microcontrollers.

EEPROM – As its name refers, it can be programmed & erased electrically. The data and program of this ROM can be erased & programmed about ten thousand times. The duration of erasing and programming of the EEPROM is near about 4ms to 10ms. It is used in microcontrollers and remote keyless systems.

4/8/23

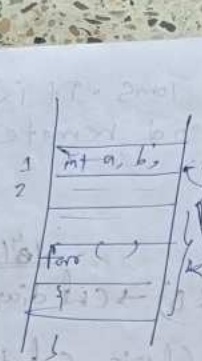
1. Design of Adder (simple, BCD) → ckt diagram
2. Carry Look Ahead Adder (logic, ckt diagram) [Lab] (Full)
3. mux, Decoders, Encoders, Comparator. [TT, ...] → [Lab] (DL – 3rd sem)

for 10

Cache memory Im Im Imp

[CPU can directly fetch ins. from the memory.
Speed of CPU = speed of cache memory
DRAM = 90ns
SRAM = 10ns → 10 times slower than CPU]

Cache memory is small & fast memory used to increase the instruction processing rate. It's operation is based on the property of "locality of reference". Analysis of a large no. of progs shows that the CPU reference to the main memory during some time period tend to be localised/confined within a few localised areas in memory. There are 2 dimentions of the locality of reference property.



1) Spatial locality (i.e. the one can be accessed immediately after current pattern)

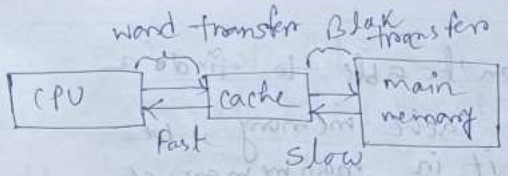
2) Temporal locality

Based on this 2 locality the cache memory works.

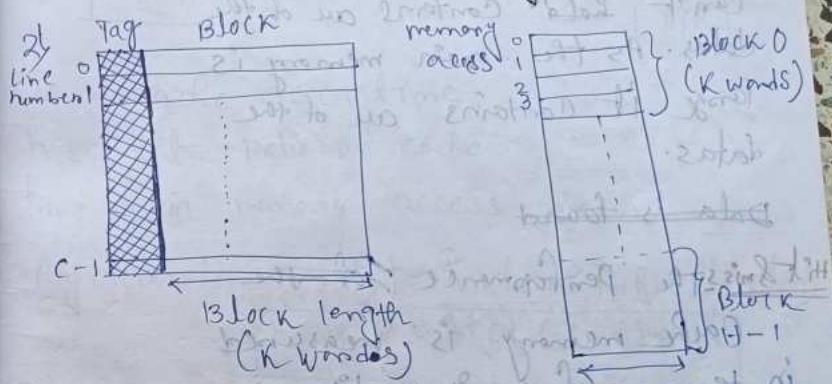
1) Spatial locality: these references to the tendency for a Prog. to access instructions whose addresses are near one another. Ex- operations on arrays, Prog. segments such as routines & macros are stored in the same memory space.

2) Temporal locality - Recently referenced instruction are likely to be referenced again in the near future. this is often caused by special Prog. Constructs such as loops & subroutine (in c it is function).

Working Principle of cache memory



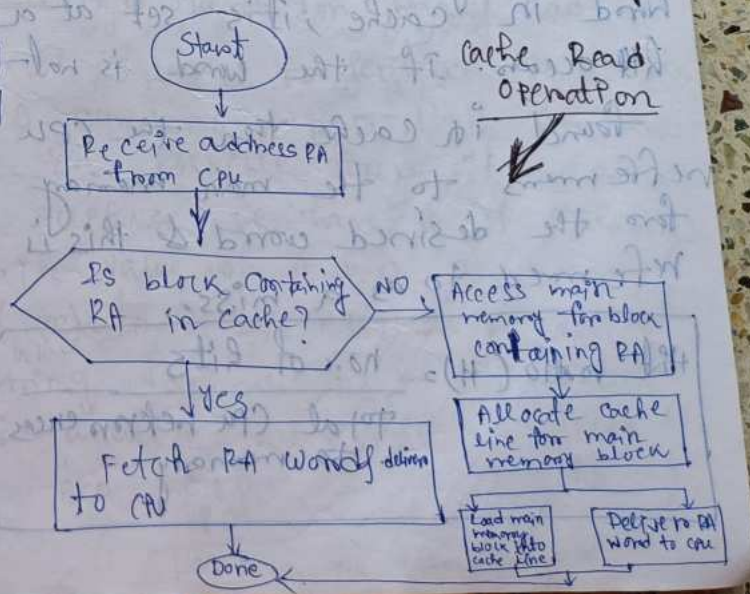
Cache & main memory organisation



[Size of line in cache = word size of main memory]

Cache

[RA = Read address]



Cache Read Operation

If we can't able to find a data in cache memory, we can find it in main memory as cache memory is small it can't hold contains all of the datas. As the main memory is large it contains all of the datas.

~~Data is found~~

Hit & miss: The performance of the cache memory is measured in terms of a quantity known as hit ratio. When the CPU references to memory & finds some the word in cache, it is set at a hit occurs. If the word is not found in cache then the CPU references to the main memory for the desired word & this is referred to as a miss.

Hit ratio (H) = $\frac{\text{no. of hits}}{\text{total CPU references to memory}}$

$H = \frac{\text{no. of hits}}{\text{no. of hits} + \text{no. of misses}}$

H is a probability, & it's range,

$0 \le H \le 1$ [H/A]

t_c = cache access time
h = hit ratio of cache
t_m = main memory access time

Avg access time (t_{avg}) = $h * t_c + (1-h) * (t_c + t_m)$

$t_{avg} = h t_c + (1-h) t_m$

1) Design of full (simple) Adder - (ckt diagram) Done

BCD Adders BCD stands for binary coded decimal. It is used to perform the addition of BCD numbers. A BCD digit can have any of ten possible four-bit representations. Suppose, we have 4-bit numbers A & B. The value of A & B can vary from 0 (0000 in bin.) to 9 (1001 in bin) as we are considering decimal no.s. There are 6 unused codes (1010 - 1111)

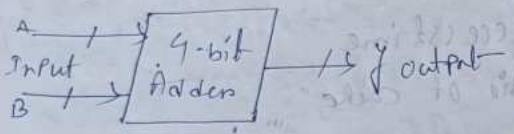
Ex- 1) $\begin{array}{r} 4 \\ +5 \\ \hline 9 \end{array}$ BCD $\Rightarrow$ $\begin{array}{r} 0100 \\ +0101 \\ \hline 1001 \end{array}$ ↳ NO error code

10_{10} 1010_{15}

11 → 4 + 8 = 12 → 0100 + 1000 → 1100 (Binary 12 → but it's invalid code) + 0110 (some have to add 6) → 00010010 → (12 in BCD)

Error code :

Symbol -

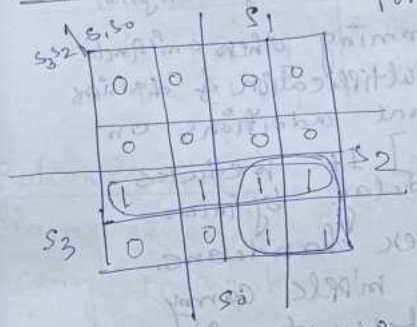


K-map Truth-table

Decimal	Binary Sum C S3 S2 S1 S0	BCD Sum C S3 S2 S1 S0
0	0 0 0 0 0	0 0 0 0 0
1	0 0 0 0 1	0 0 0 0 1
2	0 0 0 1 0	0 0 0 1 0
3	0 0 0 1 1	0 0 0 1 1
4	0 0 1 0 0	0 0 1 0 0
5	0 0 1 0 1	0 0 1 0 1
6	0 0 1 1 0	0 0 1 1 0
7	0 0 1 1 1	0 0 1 1 1
8	0 1 0 0 0	0 1 0 0 0
9	0 1 0 0 1	0 1 0 0 1
10	0 1 0 1 0	1 0 0 0 0
11	0 1 0 1 1	1 0 0 1 1
12	0 1 1 0 0	1 0 0 1 0
13	0 1 1 0 1	1 0 0 1 1
14	0 1 1 1 0	1 0 1 0 0

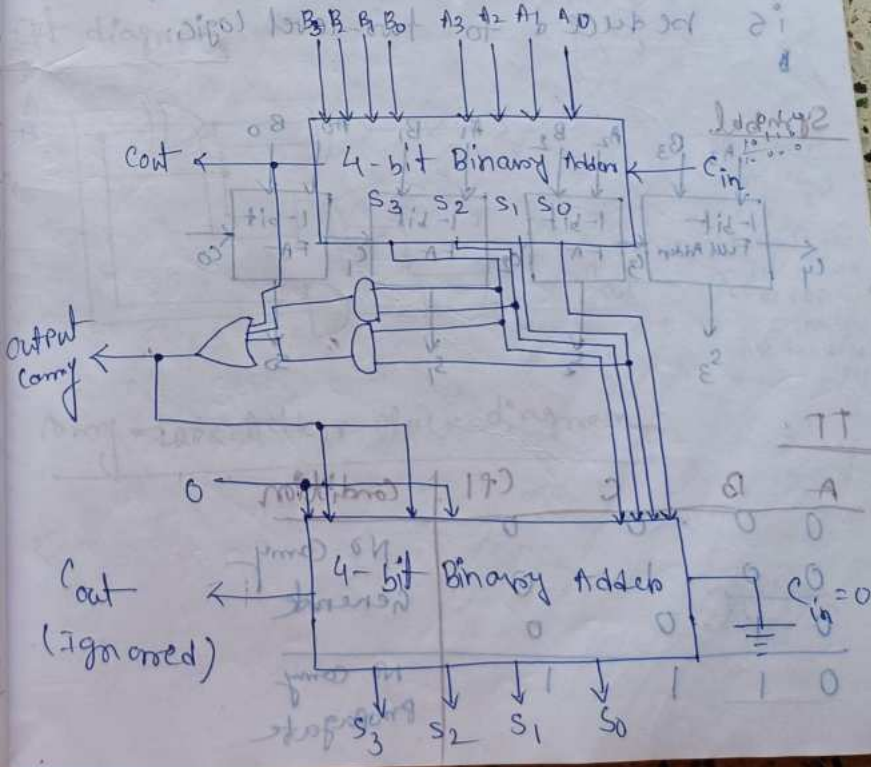
13 → 01111 → 10010

K-map for C_0



the boolean expression for carry (C) = $S_2S_3 + S_3S_1$

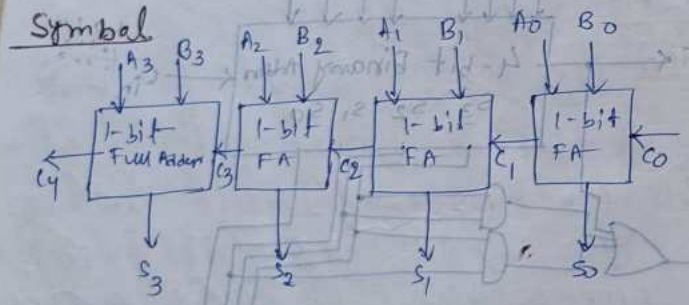
Ckt diagram



Carry Look-Ahead Adder

[The adders produce carry propagation delay while performing other arithmetic operations like multiplication & divisions as it uses several additions or subtraction steps.] It reduces the propagation delay by introducing more complex hardware. In this design, the ripple carry design is suitably transformed such that the carry logic over fixed groups of bits of the adder is reduced to two-level logic.

Symbol



TT

A	B	C	Cfl	Condition
0	0	0	0	No Carry Generate
0	0	1	0	
0	1	0	0	
0	1	1	1	No Carry Propagate
1	0	0	0	
1	0	1	1	
1	1	0	1	Carry Generate
1	1	1	1	

Boolean expression

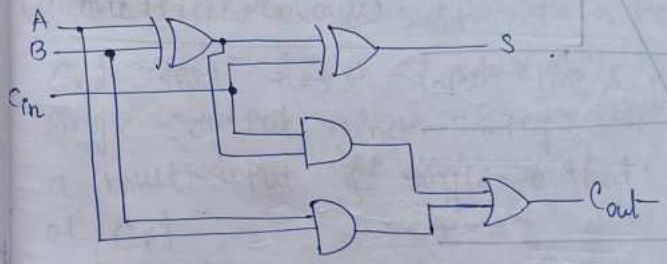
Carry generate = G_i

Carry propagate = A_i

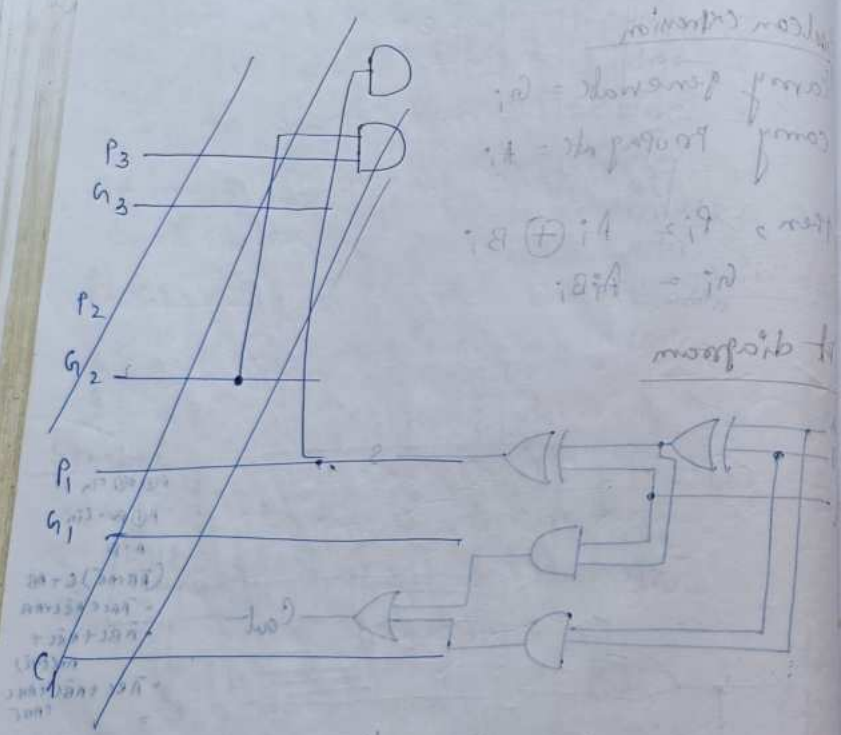
then, $P_i = A_i \oplus B_i$

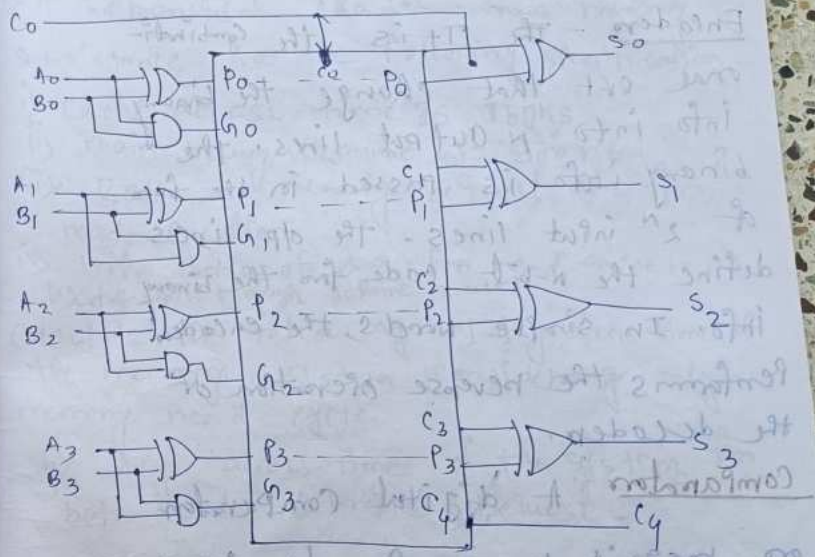
$G_i = A_i B_i$

Ckt diagram



Carry-Look Adder Block diagram





③ **multiplexer(mux)** – It is a combinational ckt that has 2^n input lines and a single output line. Simply, the mux is a multi input & single-output combinational ckt.

Decoder – It is a combinational ckt that change the binary information into 2^N output lines. The binary info is passed in the form of N i/p lines. The o/p lines define the 2^n-bit code for the binary info. In simple words, the decoder performs the reverse operation

of the encoder.

Encoders - The IT is the combinational ckt that change the binary info into N output lines. the binary info is passed in the form of 2^N input lines. the o/p lines define the N-bit code for the binary info. In simple words, the encoders performs the reverse operation of the decoder.

Comparator - A digital comparator or magnitude comparator is a hardware electronic device that takes 2 numbers as i/p in binary form & determines whether one number is greater than, less than or equal to the other number.

10/4/23

1) A hierarchical cache - main memory subsistance has the following specification -
i) cache access time is 50ns.
ii) main memory access time is 500 ns
iii) 80% of memory request are for read operation.
iv) Hit ratio of cache for read access is 90% & write through scheme is used.

Calculate the following - a) Avg access time of the memory system considering only memory read cycle.
b) Avg access time of the system for both read & write request.

t_c = 50 ns, probability of read = 0.8
h = 0.9, probability of write = 0.2
t_m = 500 ns.

$t_{avg} = t_c + (1-h) * t_m = 50 + 0.1 * 500$

$= 0.9 * 50 + 0.1 * 500 = 100$ ns

a) considering only the read cycle

$t_{avg\_read} = 0.9 * 50 + (1 - 0.9) * (50 + 500)$
$= 100$ ns

write = write back (default)
write through = write only

b) avg access time for read = 100ns

Considering both read & write cycle [formula = $P_H \cdot t_{avg\text{-}read} + (1-P_H) \cdot t_m$]

$t_{avg} = P_H \cdot t_{avg\text{-}read} + (1-P_H) \cdot t_m$ [P_H = probability of read cycle]

$= 0.8 + 100 + 0.2 + 500$

$= 80 + 100$

$= 180$ ns

[write back =
1st write cache
then, write memory
write through =
Both cache & memory

[For write through
$P_H \cdot t_{avg\text{-}rd} + (1-P_H) \cdot t_m$

[For write back
$P_H t_{avg\text{-}rd} + (1-P_H) \cdot t_{cache}$

Mapping functions

(To decide where we are going to keep a block)

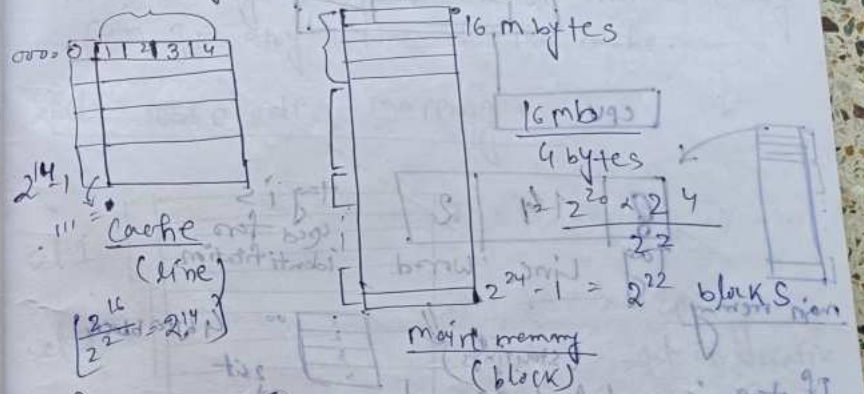
i) cache can hold 64 Kbytes ($2^6 * 2^{10}$)

ii) Data is transfered b/w main memory & cache is blocks of 4 bytes each. This means that the code is organized as $16K = 2^{14}$ lines of 4 bytes each.

iii) main memory consists of 16 mbytes, with each byte directly addressable by a 24 bit address ($2^{24} = 16M$)

Thus for mapping we will consider:

M.M consists of 4 MB ($2^{22} = 2^2 + 2^{20}$) blocks of (4 bytes each.



Direct mapping

$i = j \bmod m$

i = cache line number

j = main memory block number

m = number of lines of cache

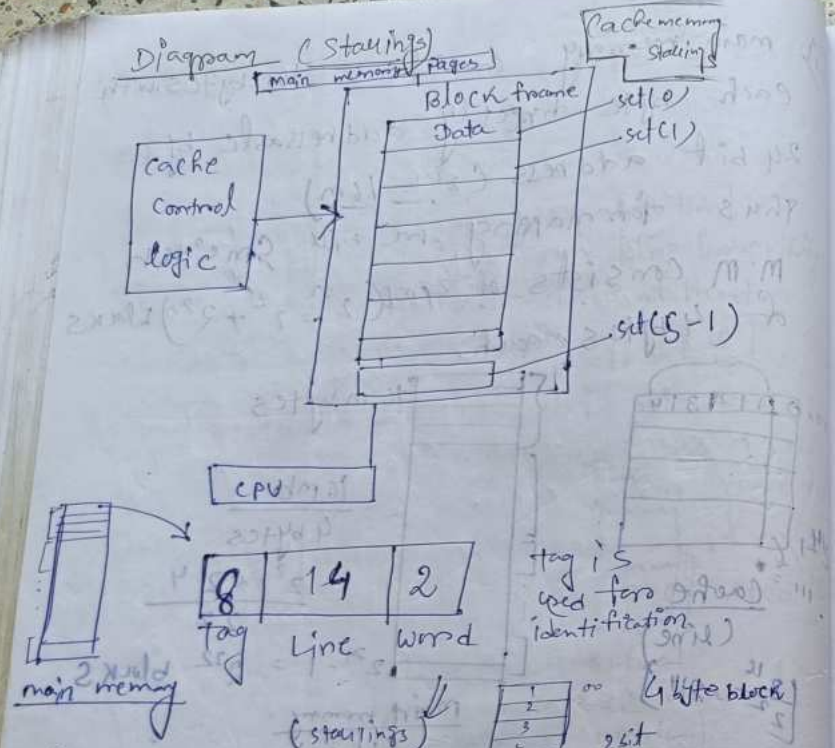
$i = 0 \bmod 2^{16}$, $16 \bmod 2^{16}$

$= 0$ $= 16$

$2^6 \bmod 2^{16}$

$= 0$

Diagram (Stallings)



tag is used for identification

(Stallings)

4 byte block

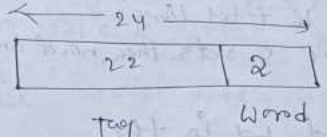
If tag is matched then the data is in, otherwise it isn't in the.

Here any block of memory can't go to the any line of the cache.

Associative mapping

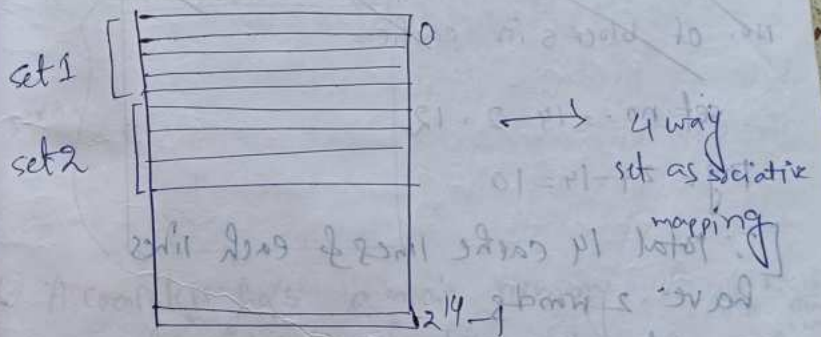
Here memory will decide it.

Associative memory



By comparing the tag we will search parallely. Any block of main memory can go any line of the cache.

Set associative mapping:



They divides the cache into sets. Outside the set it is direct mapping. Into set it is associative mapping.

$i = j \bmod v$

m = no. of lines in cache

v = no. of sets in the cache

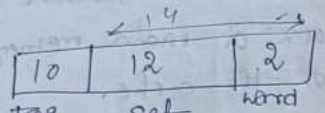
$m = v * k$

k = no. of lines within a set

K way set ~~to~~ associative mapping

[Ex- If we have 16 total lines & have ~~8~~ total 8 sets then, no. of lines in a set/~~word~~ = $\frac{16}{8}$]

we are going to a set, but in the set we can go to the any line.



24 - 14 = 10

~~No. of blocks~~ $\frac{2^{14}}{2^2} \neq m$ $V = \frac{m}{k} = \frac{14}{2} = 2^7$

No. of blocks in cache =

set no. = 14 - 2 = 12

tag = 24 - 14 = 10

[∵ total 14 cache lines & each lines have 2 words.

∴ No. sets = 14 - 2 = 12]

(another way to calculate no. of sets)

28/4/23

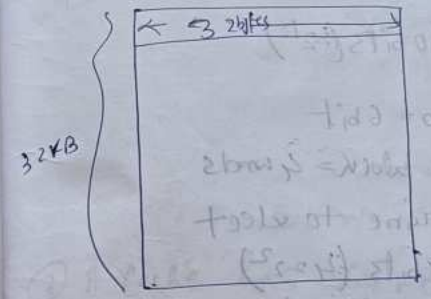
① For a cache memory of size 32 ~~bits~~ KB how many cache lines does the cache hold for block sizes of 32 & 64 - byte.

1st Case

The no. of lines in the cache = $\frac{\text{size of cache}}{\text{block size}} = \frac{32 \times 1024}{32} = 1024$

2nd Case

The no. of lines in cache = $\frac{32 \times 1024}{64} = 512$



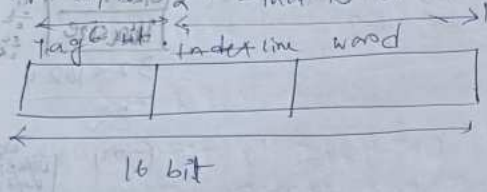
② A computer has a main memory of $\frac{64K \times 16}{\text{size}}$ & a cache memory of 1K words. The cache uses direct mapping with a block size of 4 words.

a) How many bits are there in the ~~cache~~ tag, index, ~~block~~ word & ~~what f~~ word fields of the address format.

b) How many bits are there in each word of the cache?

c) How many blocks can the cache accomodate

b) The main memory size is 64K
∴ the CPU must generate the address of $64K = 2^{16}$ that is of 16 bit

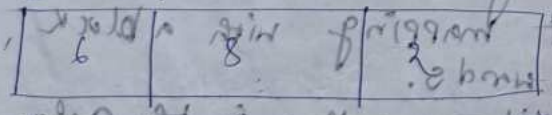


The cache memory = 1K
size

Size of the block = 10 bits ($\because 2^{10}$)

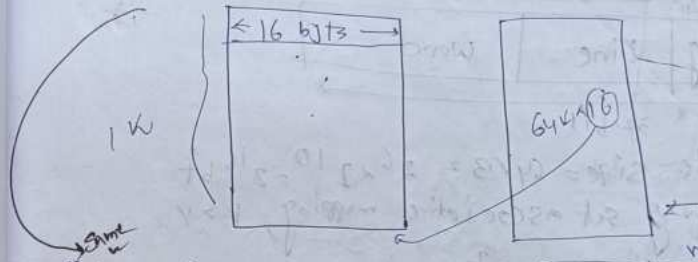
Size of the tag = 16 - 10 = 6 bit
the size of each cache block = 4 words
∴ the no. of bits require to select a word is = 2 bits ($\because 4 = 2^2$)

Size of the index field is = 10 - 2 = 8 bits



b) The main memory size is = 64K = 16 so.
the no. of bits for each words of the cache is = 16

c) from part a, we know that the line field is of 8 bits.
∴ the no. of blocks in cache is = 256



size of one line in cache = $\frac{\text{size of the cache}}{\text{size of the word}}$

$= \frac{1024}{4}$

$= 256$

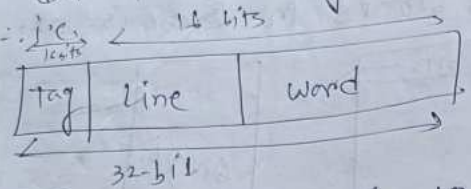
(3) A Cache has 64 KB capacity, 128 bytes lines & is four way set associative. The CPU generates 32 bit address for accessing data in the memory.

a) How many lines & sets does the cache have?

b) How many entries are required in the tag field?

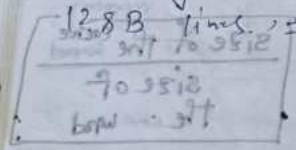
c) How many bits of tags are require in each entry in the tag field?

CPU generates 32-bit address for accessing data in the memory.



cache size = 64KB = $2^6 \times 2^{10} = 2^{16}$ bit

4-way set associative mapping, K = 4



$= 2^7 \times 8$ bits

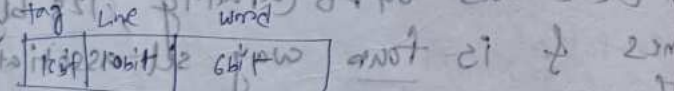
$= 2^7 \times 2^3$

$= 2^{10}$ bits

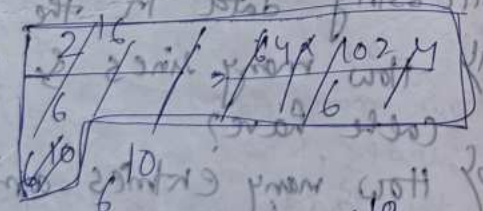
a) ~~No. of lines =~~

∴ Size of block = 16 bits

Between them 10 bits are retained for line field.



a) No. of cache lines =



~~No. of sets the cache have (v) =~~

No. of sets the cache have (v) = $\frac{m}{K} = \frac{2^{16}}{4} = 2^{14}$

No. of lines within a set (K) = $\frac{2^{10}}{2^4} = 2^6$

No. of sets that the cache have (v) = $\frac{m}{K}$

$= \frac{\text{no. of lines in cache}}{K}$

$= \frac{2^{16}}{2^4}$

$= 2^{12}$

b) the tag field is 16 bits.

∴ 16 entries are needed / retained for tag field.

$16 = 2^4$

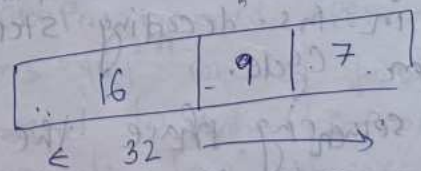
∴ 4 entries are needed for tag field

a) 4 entries ⟶ 16 bits tag

1 ⟶ $\frac{16}{4}$

$= 4$ bits

4 bits of tags are retained in each entry in the tag field.



$\frac{2^9}{2^7} = 2^2$ sets

tag = 32 − 16

= 16

No. of sets that the cache have (v) = $\frac{m}{k}$ = no. of lines in cache

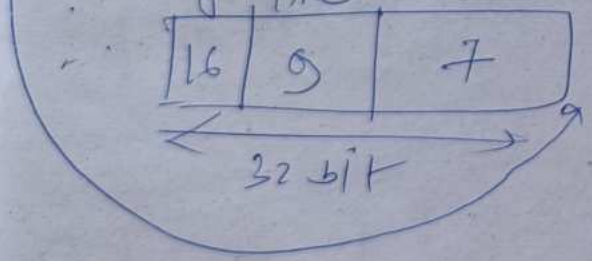
a) No. of cache lines = $\frac{\text{size of cache}}{\text{size of line}}$ = $\frac{2^{16}}{2^7}$ = 2^9 (1 line in cache = 1 word in main memory)

b) [∵ line size means (block size) = [tag - rest of part of address]]

~~9 bits = line~~

No. of sets = $\frac{2^9(m)}{2^2(k)}$ = 2^7 [∵ no. of sets = word] [4 way set associative mapping]

CPU generates 32 bit address = given



b) tag = 32 - 16 = 16

16 entries are required in tag field

Control Unit (Anand)

In Hardwared Controlled flipflop, latches etc (All sequential ckts) (..... digital ckts)

In Micro programmed controlled (Prog will run according to prog.)

Hardware controlled: A control unit performs the following responsibilities—

1) Instruction interpretation
2) Instruction sequencing

During ins. interpretation phase the CU reads ins from the memory. It then resolves the ins. type & addressing mode, gets the necessary operands & routs them to the appropriate functional units of the execution unit. Required signals are then issued to the different units of ALU to perform the operation & the results are routed to the specific destination. Thus this phase is done in ins. decoding step of the instruction cycle.

During the sequencing phase the CU finds the address of the next ins. to be excuted & loads it into the program counter, thus this phase is done in instruction fetch step of the instruction cycle. (What is CU & its work)

CU design

CU are designed into 2 different ways

1) Hardwared controlled 2) micro programmed controlled.

When the controll signals are generated using conventional sequential logic design techniques the controle unit is said to be handwared. The sequential logic ckt generates specific sequences of controle signals in response to externally supplied instructions. Logic gates, flipflops, decoders & other digital ckts are used to implement hardwared controlled organisation.

Advantages - 1) H.C.U is fasters than M.C.U.

Disadvantages - 1) The design has to be modified if a H.C.U requires changes in the wiring (wires).
2) It cannot handle complex instructions.
2) It is cheaper than M.C.U.

In microprogrammed approach all controle functions that can be simultaniously activated are grouped to form controle words & stored in a seperate long memory called the controle memory. From the control memory the controle words are fetched one at time &

the individual control fields are routed to various functional units to activate their appropriate ckts.

Advantages - 1) It can handle complex ins.
2) It provides a well structured control organisation. The control signals are embedded in a 2-level software known as firmware (mixture of hardware & software)
3) Control signals for many ins. can be generated.
4) Easy to modify as the modification needs to be done only at the instruction level

Disadvantages - 1) The m.c. approach is more expensive than the hardwired approach
2) It is much slower than hardwired approach.

Difference between H.C.U & m.c.u
↓ Used in RISC arch.
↓ Used in CISC arch.

Attributes	HCD	MCD
1) speed	Fast-	slow
2) cost implementation	more	cheapero

3) Implementation approach	Sequential circuit	Programming
4) Flexibility	Not flexible	flexible
5) Ability to handle complex instruction	Difficult	Easier
6) Design process	Complicated	systematic
7) Decoding & sequencing	Complex	Easy
8) Application	RISC microprocessor	CISC microprocessor
9) Control memory	Absent	Present
10) chip area	Less	more

prev yr in sdn 2022

Ans-A

1) How many 128×8 memory chips are needed to provide a memory capacity of 4096×16?

the no. of ram chips ~~that are~~ needed are — $\frac{4096 \times 16}{128 \times 8}$

$\frac{2^{12} \times 2^4}{2^7 \times 2^3}$

$= 2^6$

= 64 RAM chips are needed

2) Define the functions of MAR and PC.

MAR = It stands for memory addressing register. It is used to access data & instructions from memory during the execution phase of instruction. It holds the memory location of data (instruction) that needs to be accessed.

PC = It stands for Programme Counter. This register manages the memory address of the instruction to be executed next.

3) what is meant by locality of reference?

Done

4) convert the hexadecimal number 6BBE to octal.

Hexadecimal → binary / decimal → octal

$(6BBE)_{16}$ in Binary $= (0110\ 1011\ 1011\ 1110)_2$

octal

$= (064276)_8$

$= (64276)_8$

$\therefore (6B8E)_{16} = (64276)_8$

5) mention 2 features of cache memory.

~~It is highly capable of high speed.~~

1) Its speed is high

2) The progs and datas that are frequently needed ~~are~~ reside in the cache memory

3) It is small

4) It works on the basis of the Property of "locality of reference".

5) It is used to increase the instruction Processing rate.

6) Cache memory use associative access method.

Done

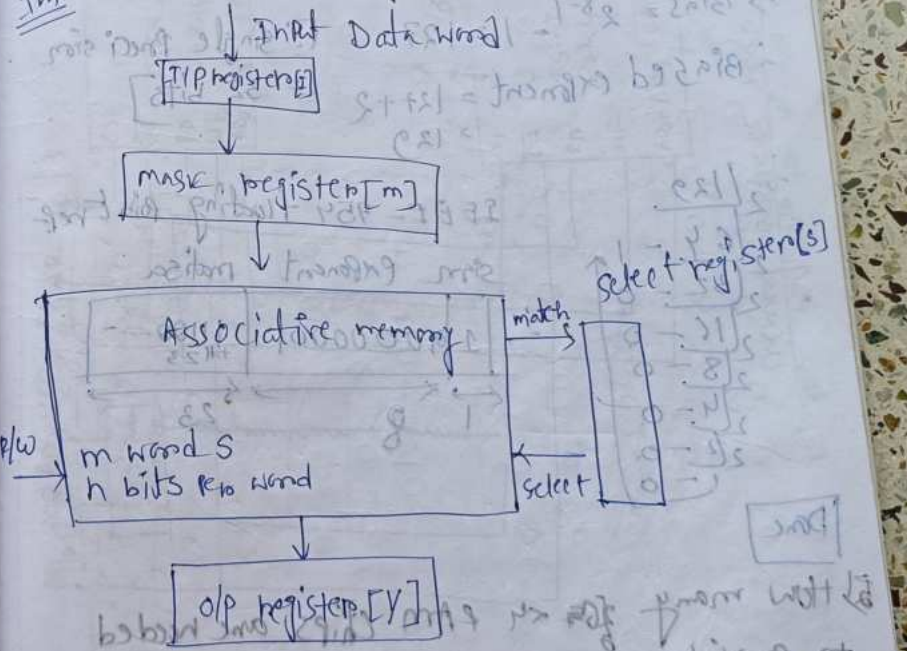
Qn-B

1) Describe the working of static RAM memories. Done

2) what is addressing modes? Explain any 3 addressing modes with suitable diagram & examples. Done

3. Briefly describe the hardware organization of associative memory. Done

Im



Block diagram of Associative memory/CAM

4) Represent the decimal value -7.5 in IEEE-754 single precision floating point format. Compare RISC & CISC architectures.

a) The decimal no. -7.5 in binary = -111.1

2 | 7
2 | 3 - 1
2 | 1 - 1

0.5 × 2 = 0 (Carry = 1)

b) The normalization form = -1.111×2^2

c) Bias = $2^{8-1} - 1 = 127$

Biased exponent = 127 + 2 = 129

[Single Precision = 32 bits]

2 | 129
2 | 64 - 1
2 | 32 - 0
2 | 16 - 0
2 | 8 - 0
2 | 4 - 0
2 | 2 - 0
1 - 0

IEEE-754 floating point representation

sign | exponent | matisa

1 | 10000001 | 11100...0 (total 23)

1 | 8 | 23

Done

5) How many 256 × 4 RAM chips are needed to provide a memory capacity of 2048 bytes? Show also the corresponding interconnection diagram.

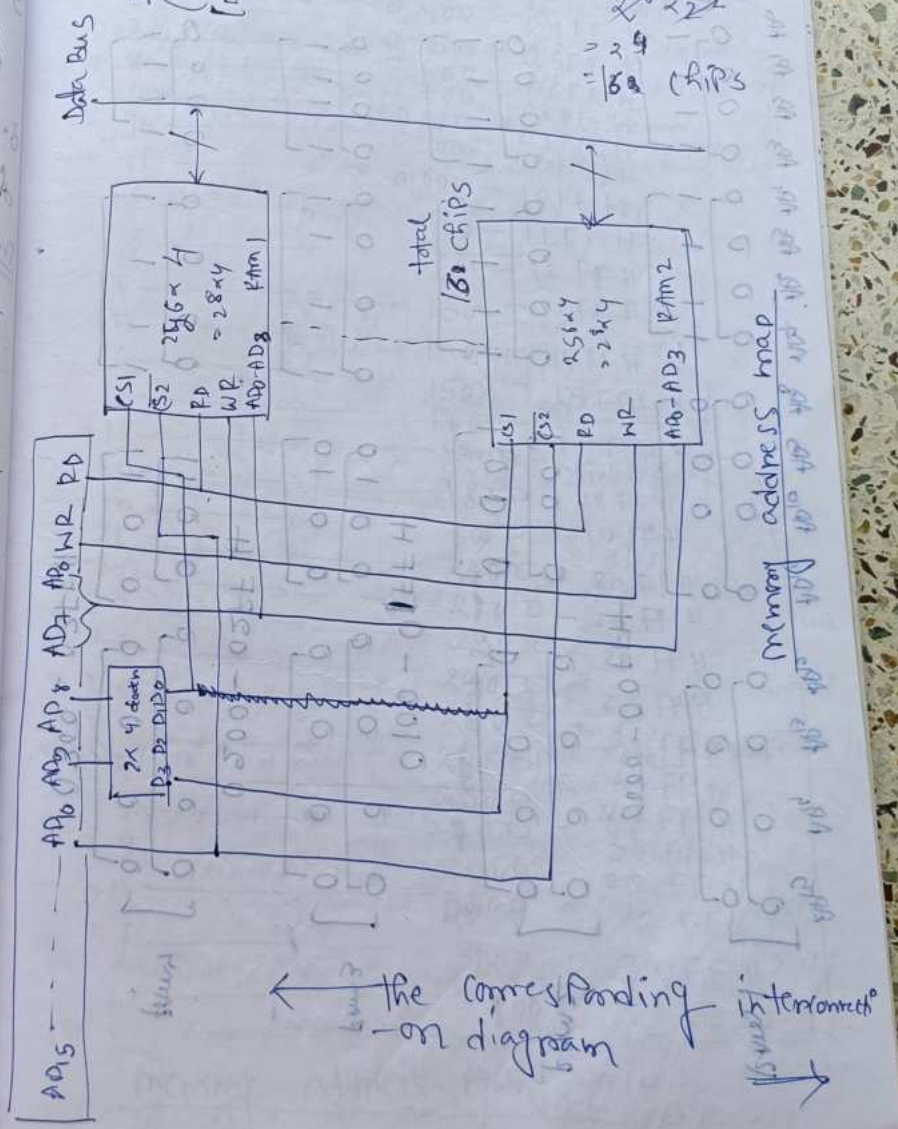
The no. of RAM chips needed are = $\frac{2048 \times 8}{256 \times 4}$

$= \frac{2^{11} \times 2^3}{2^8 \times 2^2}$

$= 2^4$

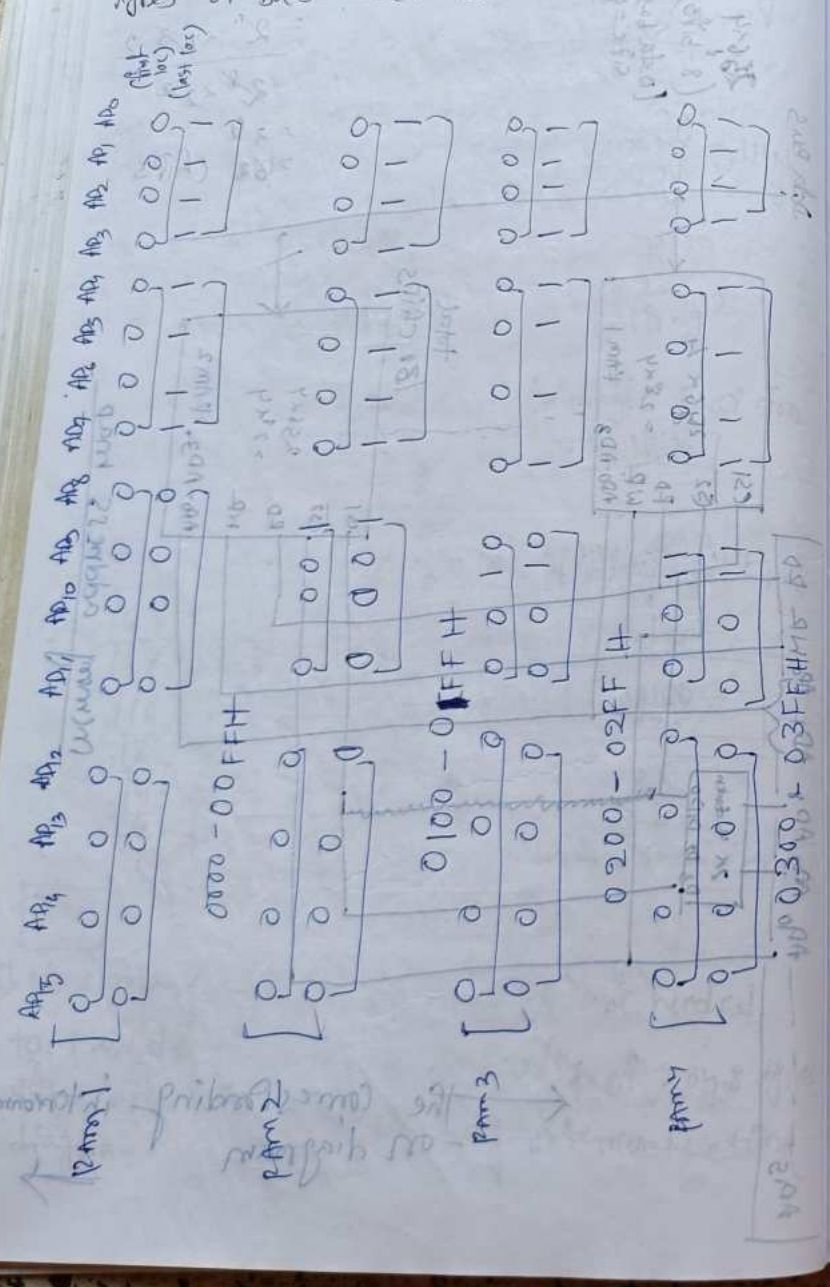
= 16 chips

256 × 4 (8-bits) [Data bus size = 4]



the corresponding interconnection on diagram

6) What is a bus? What are the different types of bus used in computers?



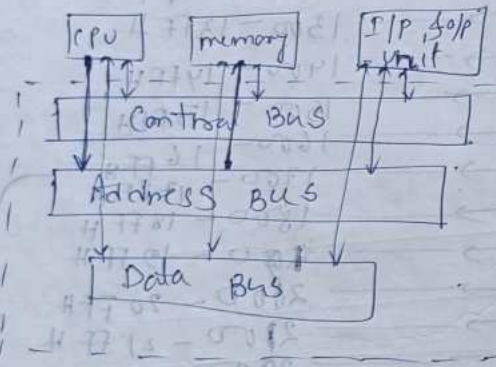
RAM 1 → 0000 – 00FF H
RAM 2 → 0100 – 01FF H
3 → 0200 – 02FF H
4 → 0300 – 03FF H
5 → 0400 – 04FF H
6 → 0500 – 05FF H
7 → 0600 – 06FF H
8 → 0700 – 07FF H
9 → 0800 – 08FF H
10 → 0900 – 09FF H
11 → 0A00 – 0AFF H
12 → 1100 – 11FF H
13 → 1200 – 12FF H
14 → 1300 – 13FF H
15 → 1400 – 14FF H
16 → 1500 – 15FF H
17 → 1600 – 16FF H
18 → 1700 – 17FF H
19 → 1800 – 18FF H
20 → 1900 – 19FF H
21 → 2000 – 20FFH
22 → 2100 – 21FF H
23 → 2200 – 22FFH
24 → 2300 – 23FFH
25 → 2400 – 24FF H
26 → 2500 – 25FF H
27 → 2600 – 26FF H
28 → 2700 – 27FF H
29 → 2800 – 28FF H
30 → 2900 – 29FF H
31 → 3000 – 30FF H
32 → 3100 – 31FF H

No need

Memory address range

Q1 What is a bus? What are the different types of bus used in computer?

Bus is a subsystem that is used to transfer data and other information between devices. means various devices in computer like (memory, CPU, I/O and others) communicate with each other through buses. In general, a bus is said to be as the communication pathway connecting 2 or more devices.



There are 3 types of bus—

1) Address bus - It is a unidirectional bus that is used to carry the memory or I/O device address, to which the data is to be transferred. the address bus in 8085 microprocessor is 16 bit wide.

2) Data bus - It is ~~bidirectional bus~~ is an 8-bit bidirectional bus that is used to transfer data between the microprocessor & other components such as memory & I/O devices. It is used to carry data to or from the memory or I/O devices.

3) Control bus - It is a bidirectional bus that is used to carry control signals between the microprocessor & other components such as memory to memory or I/O devices for performing specific operations.

Q2

1) What is von-Neumann architecture? Explain with diagram the IAS computer. What is the von neumann bottleneck.

Von-Neumann architecture - It is also called Princeton architecture. Von-Neumann proposed this computer architecture design in 1945 which was later known as Von-Neumann architecture. It is consisted of CU, ALU, registers & I/O. Von-neumann architecture is based on the stored program concept where instruction data & program data are stored in the same memory. This design is still used in most computers produced today.

Done

Done

2) what do you mean by Instruction cycle & machine cycle? Draw & explain the state transition diagram of an instruction cycle. Done

Machine cycle - A machine cycle consists of the steps that a computer's Processor executes whenever it receives a machine language instruction. It is the most basic CPU operation, and modern CPUs are able to perform millions of machine cycles per second. The cycle consists of 3 steps: fetch, decode & execute.

Done, Done

3) Evaluate the arithmetic statement $x = (A+B)/(C+D)$ in zero, one, two & three address machines. Done

4) Explain in brief the various mechanisms of mapping main memory address into the cache memory address. A hierarchical cache-main memory subsystem has following specifications (i) cache memory access time 160 ns (ii) main memory access time 90 ns (iii) hit ratio of cache memory is 0.9. calculate the following: (a) Avg accesstime of the memory system (b) Efficiency of the memory system.

Done

i) $t_c = 160$ ns
ii) $t_m = 90$ ns
iii) $h = 0.9$

a) $t_{avg} = h t_c + (1-h)(t_m + t_c)$
$= 0.9 * 160 + (1-0.9) * (160 + 90)$
$= 16 + \frac{0.1}{1} * 250$
$= 25 + 16$
$= 41$ ns

b) Efficiency of memory system = hit ration / access
$= 0.9 * 100$
$= 90\%$ - ?

5) multiply +12 & -11 using Booth's multiplication. Draw Circuit block diagram. what is the advantage of using Booth's Algorithm over other methods.

Done, Done

~~P.T.O~~

12 x -11

by Booth's algo → P.T.O

(m) 12 ↓ 1100 × (s) −11 ↓ 1011

Count	A	Q	Q_{-1}	m	
~~1st~~ Initial n=4	0000	1011	0	1100	Initial values
1st cycle n=3	0100	1011	0	1100	A ← A−m
	0010	0101	1	1100	ASR through A, Q, Q_{-1}
2nd cycle n=2	0001	0010	1	1100	ASR, A, Q, Q_{-1}
3rd cycle n=1	1101	0010	1	1100	A ← A+m
	1110	1001	0	1100	ASR, A, Q, Q_{-1}
4th cycle n=0	0010	1001	0	1100	A ← A−m
	0001	0100	1		ASR, A, Q, Q_{-1}

128 64 32 16 8 4 2 1
0 1 0 0 0 0 1 0 0
↓
1 0 1 1 1 1 0 0

(m) 12 × (s) −11

1100 0101

Count	A	Q	Q_{-1}	m	
Initial n=4	0000	0101	0	1100	Initial values
1st cycle n=3	0100	0101	0	1100	A ← A−m
	0010	0010	1	1100	ASR through A, Q, Q_{-1}
2nd cycle n=2	1110	0010	1	1100	A ← A+m
	1111	0001	0	1100	ASR, A, Q, Q_{-1}
3rd cycle n=1	0011	0001	0	1100	A ← A−m
	0001	1000	1	1100	ASR, A, Q, Q_{-1}
4th cycle n=0	1101	1000	1	1100	A ← A+m
	1110	1100	0	1100	ASR, A, Q, Q_{-1}

PW Yr 9n 2021

4p-A

1) How many 128×8 memory chips are needed to provide a memory capacity of 4096×16? Done

2) Define the function of MAR and MDR?

MAR = Done

MDR = A memory buffer register (MBR) or memory data register (MDR) is the register in a computer's CPU that stores the data being transferred to and from the immediate access storage. It contains a copy of the value in memory location specified by the memory address register (MAR).

3) What is meant by locality of reference? Done

4) Convert hexadecimal number 6 8BE to octal? Done

5) Mention any 2 features of cache memory. Done

4p-B

1) Describe the working of SRAM memories. Done

2) What is addressing modes? Explain any 3 addressing modes with suitable diagram & examples. Done

3) A Computer uses a memory unit with 256K words of 32 bits each. A binary instruction code is stored in one memory word. The instruction has 4 parts: an indirect bit, an operation code, a register code part to specify one of the 64 registers and an address part.

a) How many bits are there in the operation code part, the register code part & the address part?

b) Draw the instruction word format & indicate the no. of bits in each part.

c) How many bits are there in the data inputs & the address inputs of the memory?

4) Differentiate programmed I/O & Interrupt Driven I/O. Not in syllabus

5) What is the purpose of Control unit? Explain the differences between hard-wired control & microprogrammed control. Done

6) What is pipelining? Explain the 5 stage Instruction pipeline with timing diagram. Not in syllabus

4 b-c

1) a) Represent the decimal value -7.5 in IEEE-754 single precision floating-point format. Done

b) What is von Neumann architecture? Done

c) What do you mean by Instruction cycle & machine cycle? Done

2) a) Draw the state transition diagram of an instruction cycle. Done

b) Evaluate the arithmetic statement x = (A+B)/(C+D) in zero, one & two address machines. Done

3) a) Briefly describe the hardware organisation of associative memory. Done

b) What is virtual memory? How virtual address is converted to physical address? Not in syllabus

c) Explain any 2 page replacement algorithm in detail. Not in syllabus

4) a) Explain the various mechanism of mapping main memory address into the cache memory address. Done

b) Consider a cache consisting of 256 blocks of 16 words each, for a total of 4096 words & assume main memory addressable by 16 bit address & it consists of 4K blocks. How many bits are there in each of tag, Block / set & word fields for different mapping techniques?

Cache size = 4096 = 2^{12}

Word size = 16 = 2^4

[∴ No. of lines (or no. of blocks from main memory) = $\frac{2^{12}}{2^4} = 2^8 = 256$ (given)]

main memory is addressable by 16 bits & it consists of 4K blocks = $2^2 \times 2^{10} = 2^{12}$ blocks

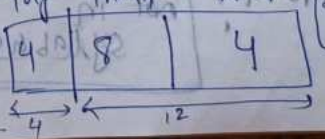
∴ total no. of blocks (words) in main memory = 4K bits

Block size of cache = 12 bits

main memory address is = 16 bits

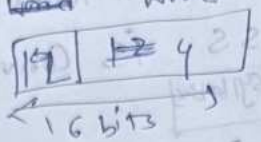
word size = 4 bits

i) Direct mapping -



[∴ tag = 16 − 12 = 4 bits]

(ii) Associative mapping:

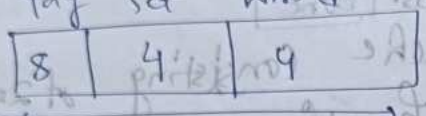


(iii) Set associative mapping:

Cache lines = 8

Word size = 4 bits

∴ No. of sets (r) = 8 - 4 = 4



(∵ tag = 16 - 8 = 8)

5) a) What is pipelining? Explain its advantage and hazards.

b) Suppose the time delays of 5 stages of a pipeline are T1 = 60ns, T2 = 70ns, T3 = 90 ns, T4 = 80ns, T5 = 50ns respectively & the interface latch has a delay of 10ns.

i) What would be the maximum clock frequency of the above pipeline?

ii) What is the maximum speed up of this pipeline over its equivalent non-pipeline counterpart?

not in syllabus