Saathi

Date 01 / 05 / 2023

ISA: Instruction set Architecture

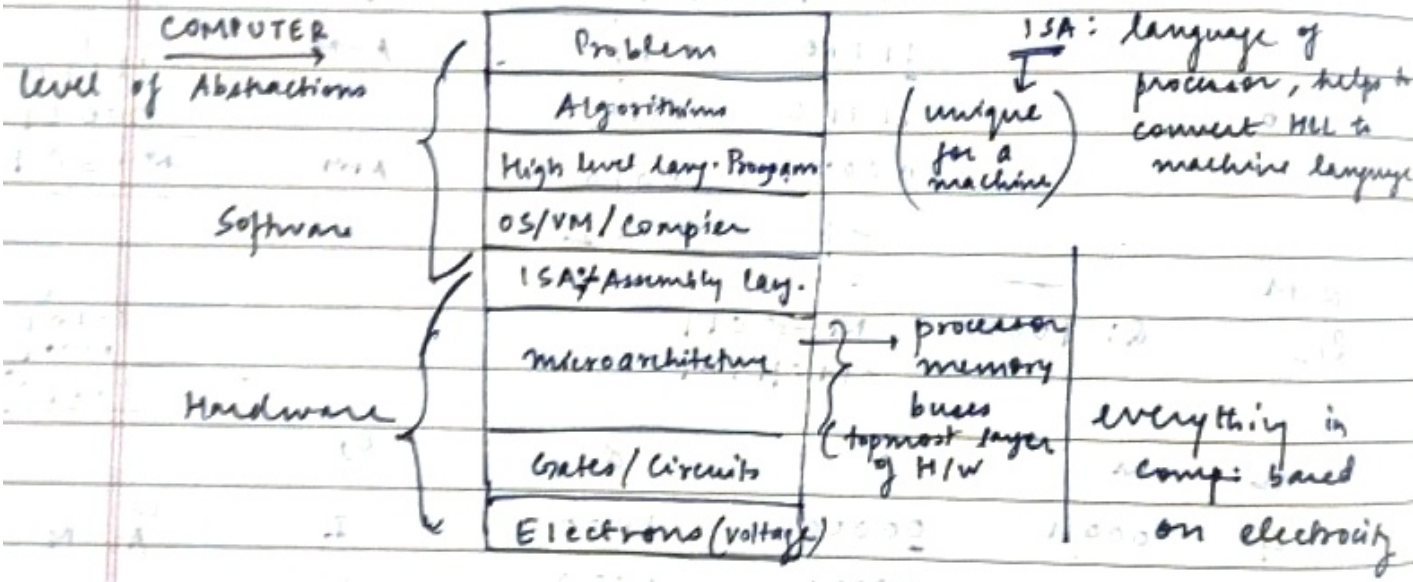
(Instruction: assembly language)

Blueprint → essential properties about the product (which does not exist) needed for the success of the product, or it will fail.

Financial → Product → User (payee → financial)

Builder

Architecture: bunch of properties decided in early phase of product development that tells whether the product has a chance at success.



Shoulders of Giants: (Brief History)

Isaac Newton: If I have seen further it is ~~because~~ by standing on shoulders of giants.

Gordon Moore: co-founder of Intel

Kathleen Booth: came up with assembly lang.

Federich Faggin: Intel 4004: first commercial microprocessor architecture of 4-bit arithmetic. Chip used in calculator.

Page No.

Date __/__/____

Saathi

Quantitative approach To design & evaluation of computer architecture : Henny & Peterson

RISC → ISA

↓

made it open source

① Number System

$(1\,0\,0\,1\,1\,1\,0\,1)_2 \rightarrow (X)_{10}$

$7\ 6\ 5\ 4\ 3\ 2\ 1\ 0$

$2^7 + 2^4 + 2^3 + 2^0 + 1$

128
16
8
1
153

$(101)_4$ → 101

$1\ 2\ 3$

100	104	108	112	116	120	124
2	3	5	1	11	0	...

Array : can directly jump to any location & do an operation (freedom)

② Stack

Stack : no freedom

So why use stack?

Organisation of data in stack simplifies the problem in several cases.

(algorithm's complexity might change)

→ variable : Top

Top = −1

Push (2)

Pop (2) : decrement top by −1 again.

Da Polish Notations:

③ Postfix: (2+4) → (24+)

very easy for a machine that uses memory as storage to solve problems.

Prefix: (+ 24) operator present b4 operand

④ Organisation & Architecture

⑤ Syntax & Semantics:

⑥ Language: ~~set of words~~ {words}

If not present in this set, then not a part of language.

= {alphabets} (26) english

= {0, 1} binary (base=2)

= {0, 1, 2} (base = 3)

Syntax (structure) error: missed something essential in program (eg: ;)

4 = a + 2; ~~two~~ wrong (∵ LHS should have a variable)

X = a + 2;

Semantics: whether it is meaningful or not

(whether it has meaning or not)

Date ___ / ___ / ____

Saathi

MUX :

Decoder

Flip flop

Registers

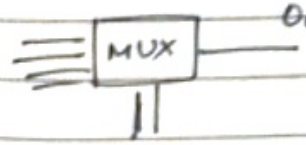
A = 0 ⇒ X = 0

(no need to know B)

X = 1 if A = 1

X = 1 if A = 0

$\bar{A}$ = 1 if ⇒ X = 0 $\quad \overline{A+B} = \bar{A}\bar{B}$



Circuits such as these use MUX: multiplexer.

If o/p = 01 ⇒ I_1 displayed

Control Input (can generate a no. of 2 bits ⇒ max. 4 channels ∴ 4 i/p)

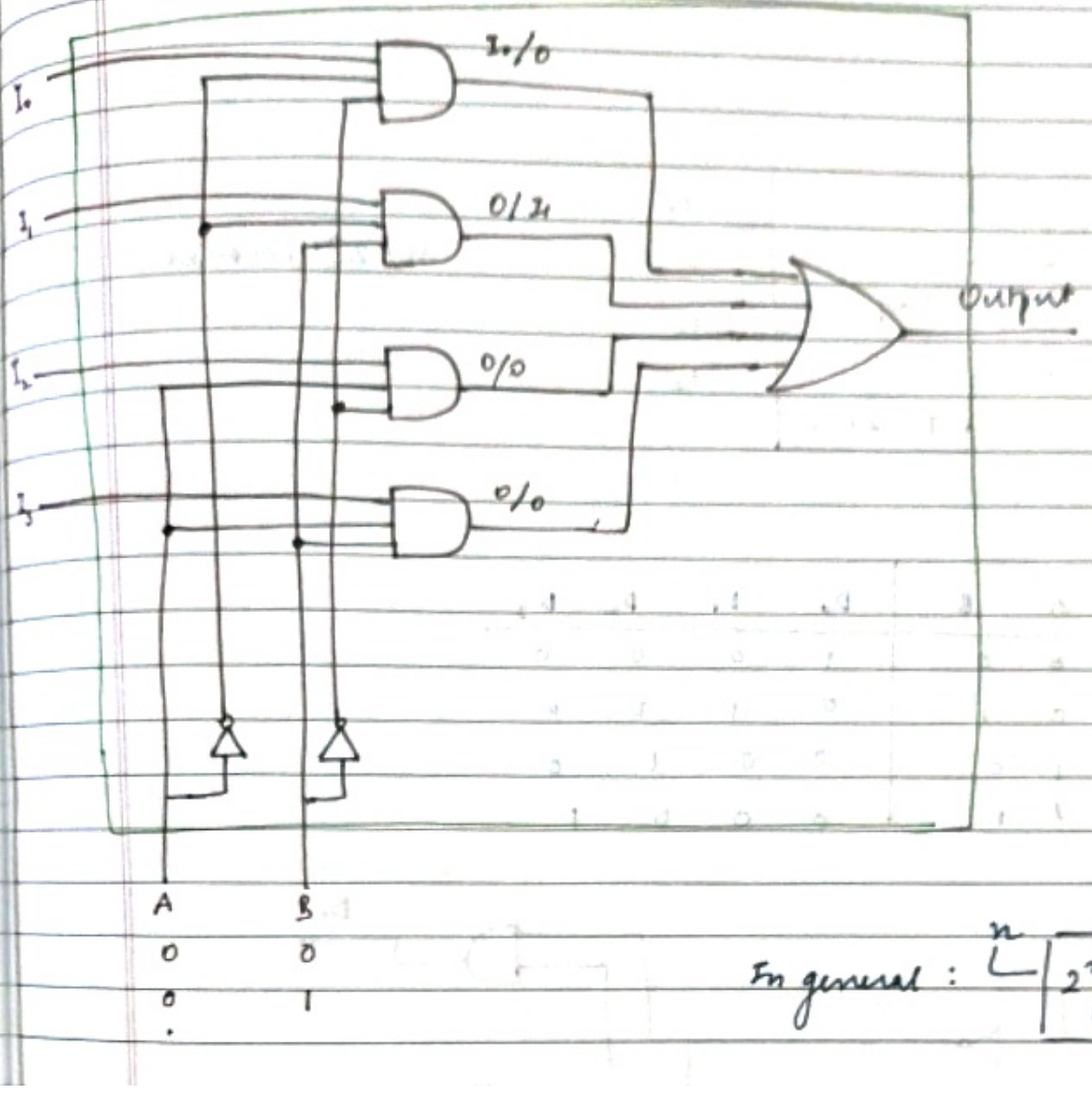
(If n bits, then we can generate 2^n diff. inputs)

Page No.

Date ___ / ___ / _____

Saathi

MUX

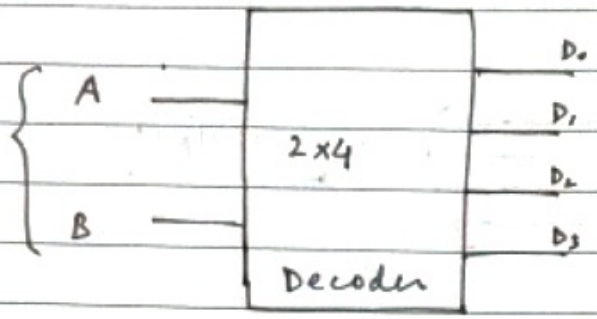


In general : n → $2^n \times 1$ MUX

Date ___ / ___ / _____

Saathi

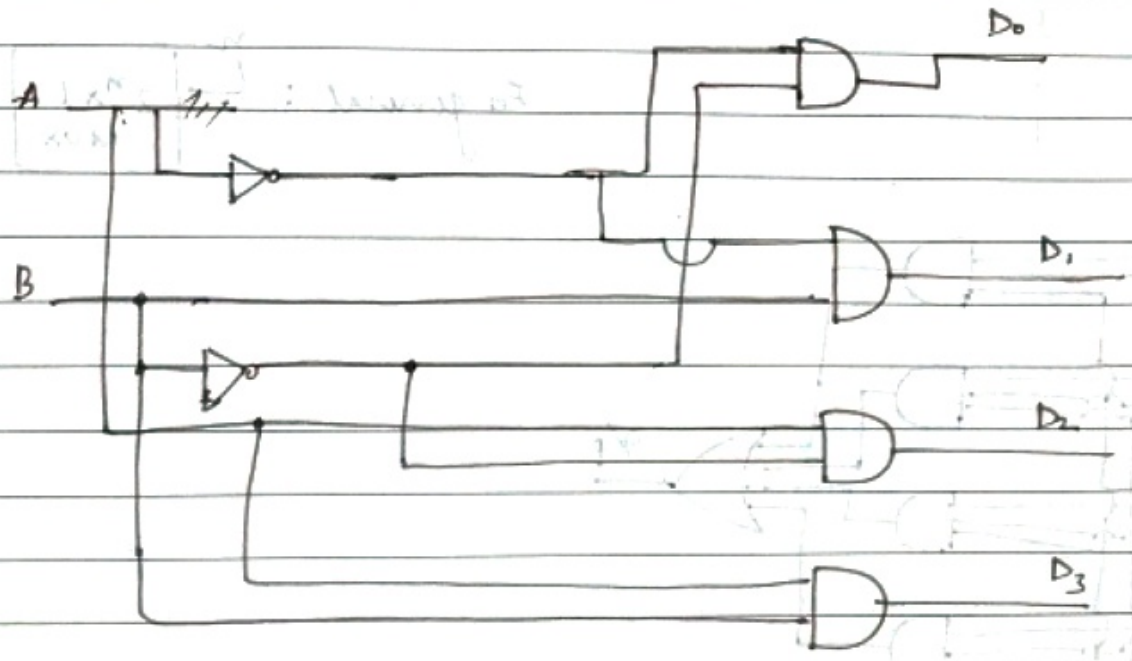
DECODER



$n \times 2^n$ decoder

Required:

A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

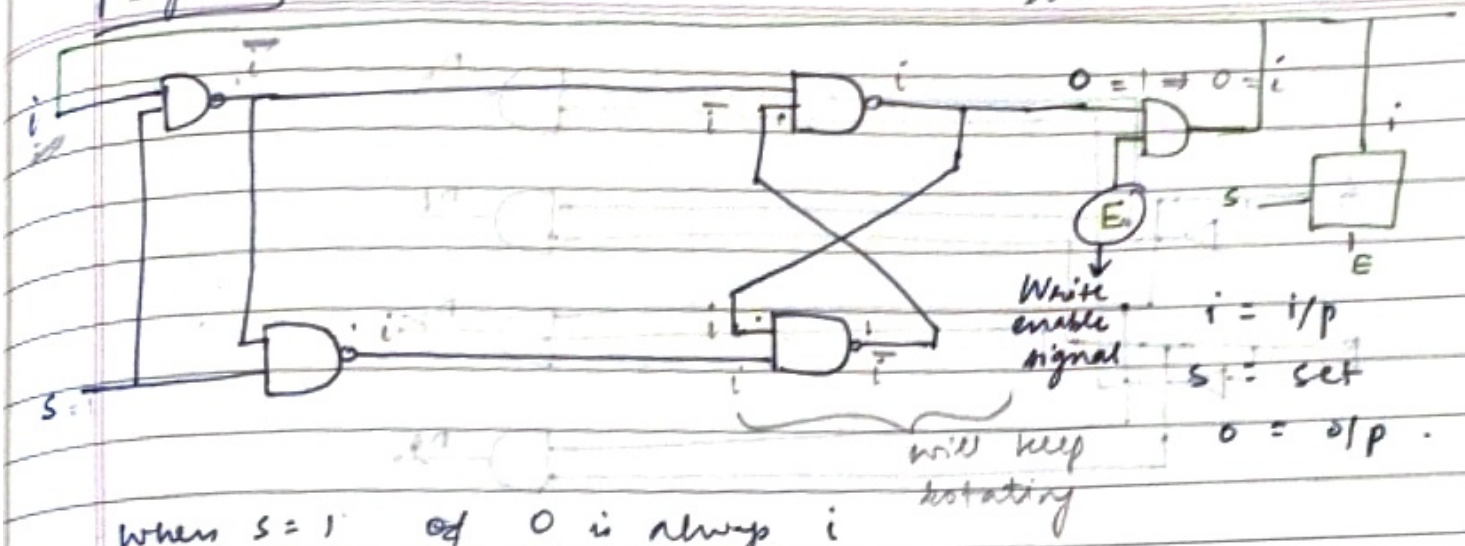


Gives $D_0 = 1$ if $A = 0$ & $B = 0$

Page No.

Register

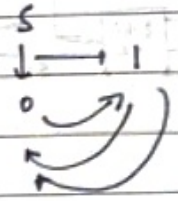
Circuit : 1 bit



When S = 1 o/p O is always i

S = 0 ~~O = zero~~ O = zero

So o/p will remain i as long as S = 1 & power is on.



bus : take data from one place & transport it to other.

when E = 0 → zero

E = 1 → output o

↓

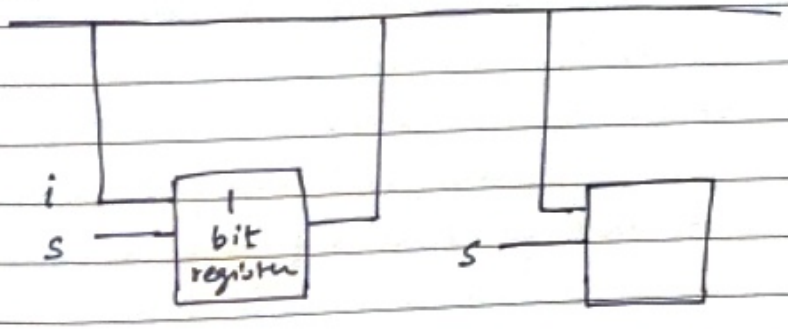
value finds its way to the bus

↓

prevents value to go to the bus.

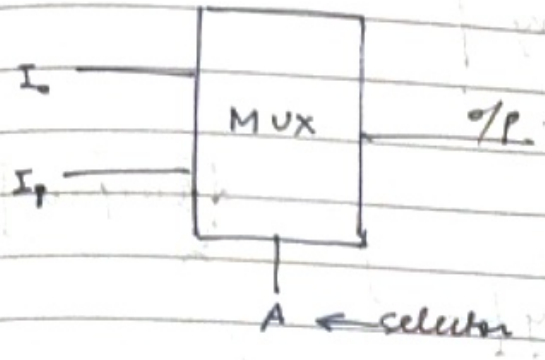
For 1 byte ⇒ 8 registers.

Bus :



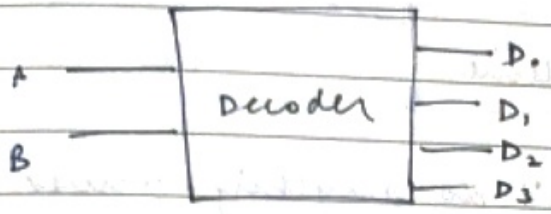
Every ~~up~~ up has a manual that explains its ISA.

Date 08 / 05 / 2023



1 bit ; 2 i/p. ; 1 selector.

If 2 bit ; 4 i/p ,, 2 selector



n i/p & 2^n o/p.

ISA : High level lang. ⟶ i/p for compiler
Assembly
Machine lang.

| 0 1 | ⟶ o/p of compiler.

Alan Turing : British scientist in WW II.
Enigma : German machine : transmitted encrypted code. So decoding required.

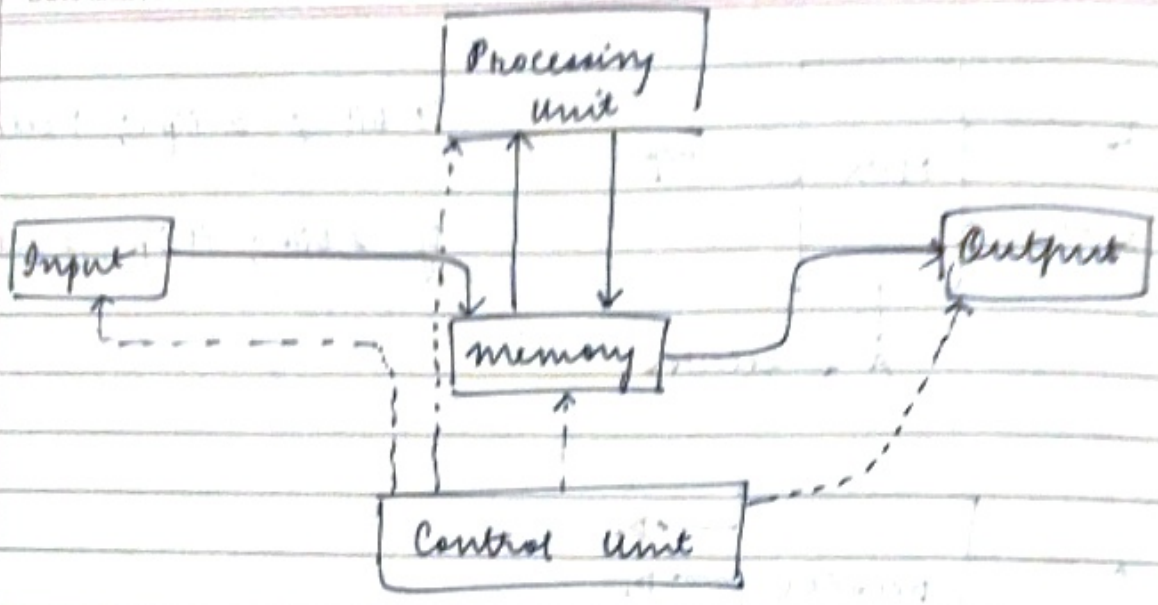
Stored Program Computer:
→ model was proposed by Von Neumann
→ Computer's memory would have program
→ Instructions would be executed sequentially.
→ All modern computers are based on this model (slight variant)

Program ⟶ Memory → Processor → Instructions
reads stored

Data flow parallel architecture structure.

Page No.

5 components of a computer



Any of the 4 parts can't do anything unless control unit sends a signal to that task is performed in a synchronised way.

Memory Component:

Size & address:

Every byte has an address. Such a memory is called byte-addressable

If memory has 16 bytes
How many bits required to uniquely identify each byte?
Total addresses = 16 (if byte-addressable memory)

= 0 15

15 = 1111
⇒ min. 4 bits required to generate 16 addresses

Memory of 2^n memory locations & each loc. has m bits
→ It has address space of 2^n uniquely identifiable locations
→ addressability of m-bits.

MAR - memory address register
MDR - " data "

Date __/__/____

Saathi

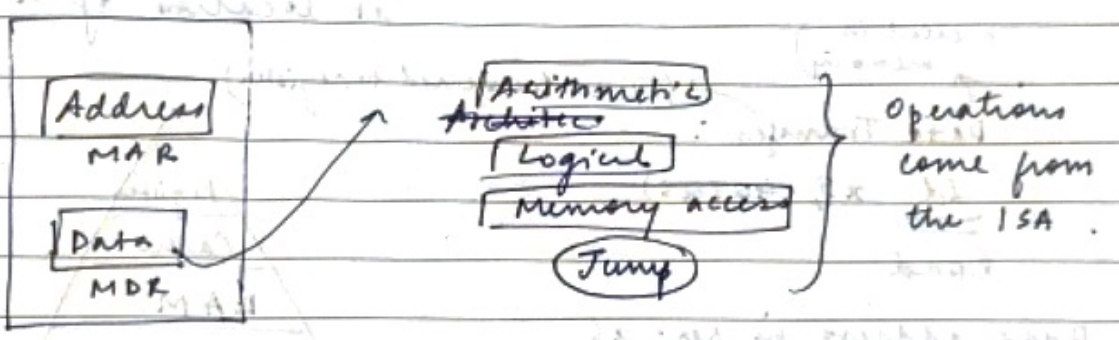
1 byte = 8 bits

4 bit → nibble

If each address has 4 bit, addressability = 4
" " " " 8 bit " = 8 bits
= 1 byte
⇒ byte - addressable memory

Read - Write:

1 read → 2 steps → MAR
→ MDR
{at atomic level}



array[i] ⇒ *(array + i * n)
base address
size of data type.

RISC V architecture
LC 3 architecture (includes 16 instructions)

add x4, x5, x6 → $x_4 = x_5 + x_6$

Operation code (op code) | operands

$x = a + b$

+ & = : operators
x, a, b : operands.

addition using 3 registers. (first register = x_0)

sub x4, x5, x6 ⇒ $x_4 = x_5 - x_6$

addi x4, x5, 20 // add immediate with constant

→ $x_4 = x_5 + 20$

Page No.

Date ___/___/_____

addi x4, x5, 10 ⇒ x4 = x5 − 10

i ⇒ immediate

C - code

a = (b+c) − (d+c)

~~add a,~~

~~sub a, add b, c, add d, e~~

↓

RISC V Assembly

add x1, x2 (d), x3 (e)

add x4, x2 (b), x3 (c) ← reuse them as registers are free.

sub x5, x4, x1

st x5, . . . // instruction to store whatever in x5 at location of 'a' (i.e. memory)

(present in memory)

Data Transfer : → (to be transferred to register)

ld x5, 40(x6)

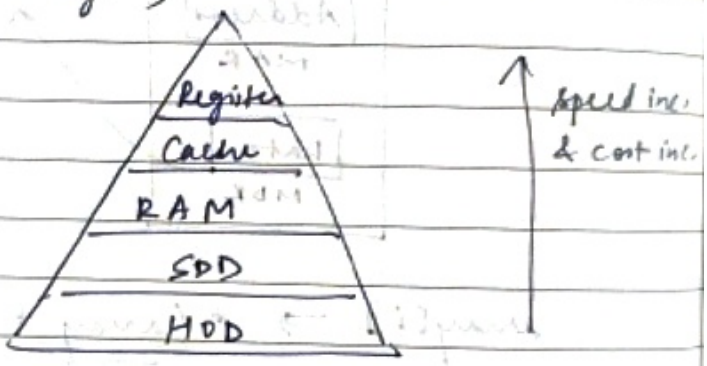
load

Base address in reg. x6

& 40 is offset

A[40]; ← array

Wherever x6 is pointing, move 40 units ahead & get the data present there into x5.



store

A = b + c ⇒ 4 instructions in RISC V = 1 instruction in C

ld ld

add

sd x5, 40(x6)

store data

Whatever is in x5, put it in the location ~~to the~~ 40 units ahead of x6.

Memory[x6+40] = x5

store doubleword

Page No.

Date __/__/____

beq x5, x6, 100 if (x5 == x6) go to PC + 100

↓ branch equal to

↓ Program counter has address of instruction which shows be executed next.

bne x5, x6, 100 if (x5 != x6) go to PC+100

branch not equal to.

bge x5, x6, 100 if (x5 >= x6) go to PC+100

↓ branch greater than equal to.

---

add x3, x4, x5 // instruction represented in binary format (executable form)

↓

add x3, x4, x5

← has 32 bits, max. 3 registers, 1 opcode

⇒ 8 bits assigned to each.

If 16 bits then opcode → 4 bits.

Each

x = A B + c use an opcode & it will be computed in 1 step. [separate steps for addn & product not required]

Cont'd

0001	0001	0010	0011
4bit	4bit	4bit	4bit

: Instruction set has 16 bits.

add x1, x2, x3 ⇒ 4 entries

↓

using 4 bits for opcode & each register

⇒ 16 different opcode possible

0001 → add ← Representation in machine language

add 0001

addi 0010

sub 0011

mul :

div :

bne :

:

Using 4 bits we can have 16 registers:

x0 ← 0000

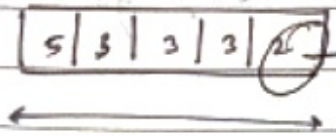
x1 ← 0001

:

x15

Date __/__/____

Case 2



→ can support new application

bits
4 ⇒ 16 opcs
⇒ for 18 opcode ⇒ 5 bits

∴ 1 reg. can't be represented by 4 bits
e.g. then 3 reg ⇒ 12 bits
5 + 12 = 17 > 16 bit instruction
X.

Case 1 → 16 registers
Case 2 → 8 registers

For a large expression with multiple variables, so case 1 is faster as it has registers (so load & store not required)
⟹ costly operation.

No. of registers
32-bit register
⇒ it can hold 32 bit data

Depending on complexity of code, either of the 2 cases could be faster.
Case 2: introduces 2 opcode at the cost of 8 registers

Case 1: used all the bits. If a new kind of data generated, no space left to add that.
So there is no scope for extension here

Case 2: There is scope for extension here.

Date __/__/__

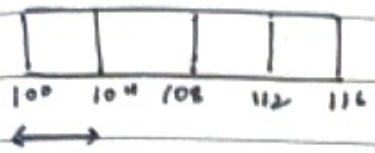
Saathi

05.23

C2 Lecture 3

Memory of size : 16 MB
Addressibility is 4-bits (data unit = 1B (each 1 byte can 1 address))
bits required for an address for this memory.

int list [4]



x = list [3] = *(list+3)
= *(100+3)
= *(100+4·3)
= *(112)

1 kilo = 1024 = 2^{10} ≈ 10^3
(now called kili)
2^{20} = Mega
2^{30} = Giga

16 MB = $2^4 \times 2^{20}$ B = 2^{24} B
Unique addresses required = 2^{24}
∴ no. of bits required = 24
(to represent all addresses)

If data unit = 4 bits ⇒ 1/2 byte
Size ⇒ 1 byte = 2 addresses.
Unique addresses required = 2×2^{24}
= 2^{25}

2^{24} B = $2^{24} \times 2 \times \frac{1}{2}$ B
(1/2 unit)
= 2^{25}

no. of bits required = 25

addressibility = 2 bytes
⇒ 1 address = 2 bytes
Unique address = 2^{24} B
= 2^{23} (2B) ∴ 23 bits req.
(1 unit)

Page No.

ISA : interface b/w programmer & microarchitecture

Process
(cache
memory)
Registers
control

ISA : manual that includes : ① Instruction set
② General Purpose Registers
③ Memory

PC ← to hold an address
IR ← Instruction register

Von Neumann Model :
→ Instructions stored in memory & executes sequentially

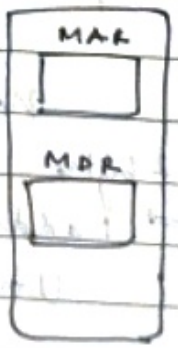
Dataflow architecture (?)
[no scope of parallelism

Byte addressibility = one

$I_1 \Rightarrow 100$ ← PC
$I_2 \Rightarrow 102$
$I_3 \Rightarrow 104$
$I_4 \Rightarrow 106$

executed one after another

Memory → MAR, MDR



Date ___/___/_____

Saathi

add x_0, x_1, x_2

opcode — destination — source

has a value which is in memory

$x = a + 10;$ ← in C

addi $x_1, x_2, 10$ ← in assembly lang.

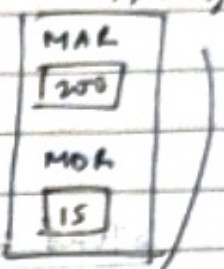
load data ~~a~~ to register before executing.

x_0	x_1	x_2	x_3	x_4	x_5
	25	15		500	200

Ld $x_2, (x_5)$

addi $x_1, x_2, 10$

sd $x_1, (x_0)$



→ PC points to next instruction.

Neural networks (non-linear)] Deep Learning

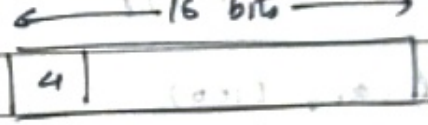
GPU — games — graphics

r, g, b (0 – 255)

more pixels ⇒ sharper image

Hardware → has a vision based on the customers targetted.

Software → Hardware → ISA



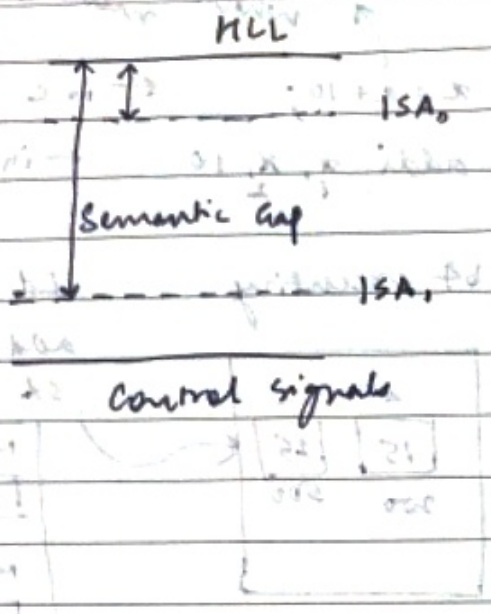
16 bits/instruction

⇒ 4 bits for opcode

8 GPRs. ⇒ 3 bits required for each.

2s comp : Negative | Positive

$(x+1)$ 0 x



If ISA is close to control signals, hardware has to simplify it for processor to understand.
(translation)
smaller semantic gap ⇒ easy for processor to process
Larger s.g ⇒ compiler has to break it to simpler parts

STORE RESULT: How many ways?

→ Initialise a variable name
→ Pointers to integer
→ Pointer to a pointer to an integer
⋮
→ Array.
→ Option to specify memory address

ld x_1, (100)
ld x_1, (x_2) ⇒ $x_1 \leftarrow$ mem [reg [x_2]]
ld x_1, 10(x_2) [x_2+10]

ISA closer to HLL has more addressing modes.

Date __/__/____

Saathi

12.09.2023

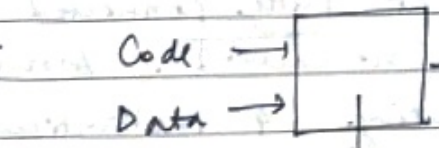
MEMORY

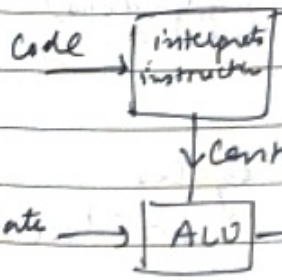
Why memory? To store instructions, data.
Primary purpose of comp → to compute/process data. So imp. to store data.

Eniac: first electronic computer (vacuum tubes)
↳ based on decimal (rather than binary)
↓ digital
using 10 vacuum tubes representing each decimal digit.
disadvantage: ① laborious & tedious work.
② Programmer would have to repeat the same sequence of operations to get the output again.

could not use this data again

If the device is used for same type of computation, then we don't need memory to store the instructions. Hardwire the Programs logic into the hardware itself. (∵ Code not required)
If the device is used for any type of computation, then code is required with data to give output.





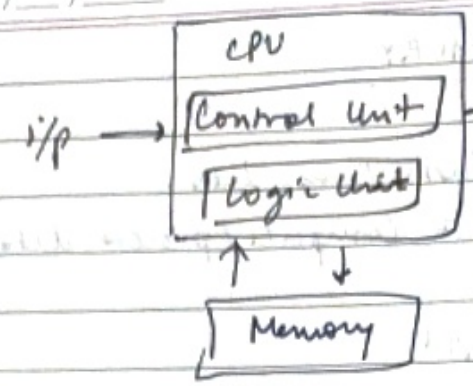
Control Unit: generates control signals
CU + ALU together form the processor

Every time we have to give the code as i/p to repeat the operation. So we need to store the code in memory.
Then came von Neumann Architecture. (around 1940s)

Page No.

Von Neumann Architecture

Date __/__/____



← Classical comp. today are based on it.

Code

I_1 –
I_2 –
I_3 –
I_4 –
I_5 for i = 1 to 100
I_6 a[i]++ = 3;
I_7

load addres to the program counter.
↓
Stores the address of the next instruction to be performed.

Memory has not evolved at the same pace as processor.

Memory Latency:

Consider a processor operating at 1 GHz (→ 10^9 clock cycles per second) connected to a DRAM with a latency of 100 ns (no cache). Assume the processor is capable of executing 4 instructions in each cycle of 1 ns.

Peak processor rating is? 4 GLOPS (gigaflops)

4×10^9 instructions per second.

flops: floating point operations per second.

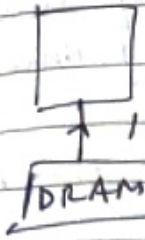
1 sec ← 10^9 cycles
1 cycle → 10^{-9} sec.
= 1 ns.
↳ nanosecond

Page No.

Date / /

Saathi

Processor



100ns to fetch data from DRAM

Processor cycles are being wasted
∵ it has to wait
In 100, it could've completed 400 cycles.

Key characteristics of comp - memory system.

Internal : directly ~~accessibly~~ accessable by processor

External : not " " " "

Main memory → Volatile
Non - ~~ea~~ erasable → ROM.

Memory hierarchy → how much? how fast? how ~~expensive~~. expensive.

Registers : fastest
smaller ~~cache~~ Cache is better than ?
↓
faster than main memory
made up of gates they have ~~no~~ ~~of~~ gates)

Memory Hierarchy

cost ↑
size ↓
speed ↑

SSD : 512 GB
↓
Main memory → 8/16 GB
> SSD.
∴ SSD lies between main memory.

Principle of locality

Page No.

Saathi

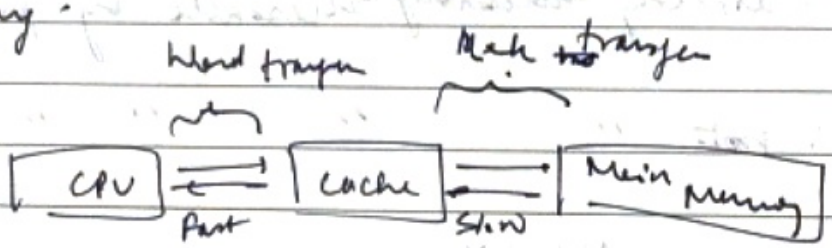
Date __ / __ / ____

Principle of Locality: program access a relatively small portion of the address space at any instant of time.

Refers to time → Temporal Locality: keep most recently accessed data items closer to the processor.

Refers to space → Spatial locality: Move blocks consisting of contiguous words closer to the processor

Access from cache is faster (approx. 5 times) than main memory.



Single cache

3 level cache organisation.