Prepared and Written By
RAZEEN AHMED

CSE-340

[COMPUTER ARCHITECTURE]

HANDWRITTEN NOTE

Computer Science and Engineering
BRAC University

Github: github.com/razeen
LinkedIn: linkedin.com/in/razeenahmed

Subject : Date : 02/06/24
Time :

Chapter-1

In MSP → from every chapter there will be 3-4 marks of questions.

Eight Great Ideas :-

We have to optimize the common case.

To understand the common case we have to perform ~~an~~ some experiment.

☆ We achieve pipelining through parallelism

Abstraction :-

Hide the lower level details to keep things simple.

Performance via prediction :-

If we can predict ~~a~~ a certain incident before its occurrence then we can save some amount of time.

Depedibility via ~~dependency~~ redundancy :-

two types
- → Software redundancy
- → hardware "

→ we add some extra components in our system

Example :-
→ we ~~can~~ keep extra tire in our car
→ we keep pressure sensor in our car.

Subject : MID Chapter → 1 + 3 → math based question.

Date :
Time :

Levels of Program code

High level language
↓
Assemby language
↓
Machine language
[Hardware representation]

Touch screen

→ Resistive (industry machineries, medical devices, automotive apps etc)

→ Capacitive
(Tablets, smartphones)

Datapath:

(i) Data
(ii) Processor
(iii) Instruction
(iv) Data flow
(v) Functional unit

If we compare the "DATAPATH" To a "KITCHEN"

(i) Data → Ingredients
(ii) Chef → Processor
(iii) Kitchen tools → Functional units
(iv) Recipe → instructions
(v) Kitchen workflow → Data flow

Kitchen:-
data → ingredients
chef → processor

up to $\boxed{\text{Slide-15}}$

ISA → ~~It defines~~ Different ISA's defines "how" a processor understands and executes instruction.
(Instruction set Archetecture)

ABI → It defines how software interacts with the System's hardware and OS.
(Application Binary

Subject :

Date :

Time :

DATAPATH :-

(i) ISA :

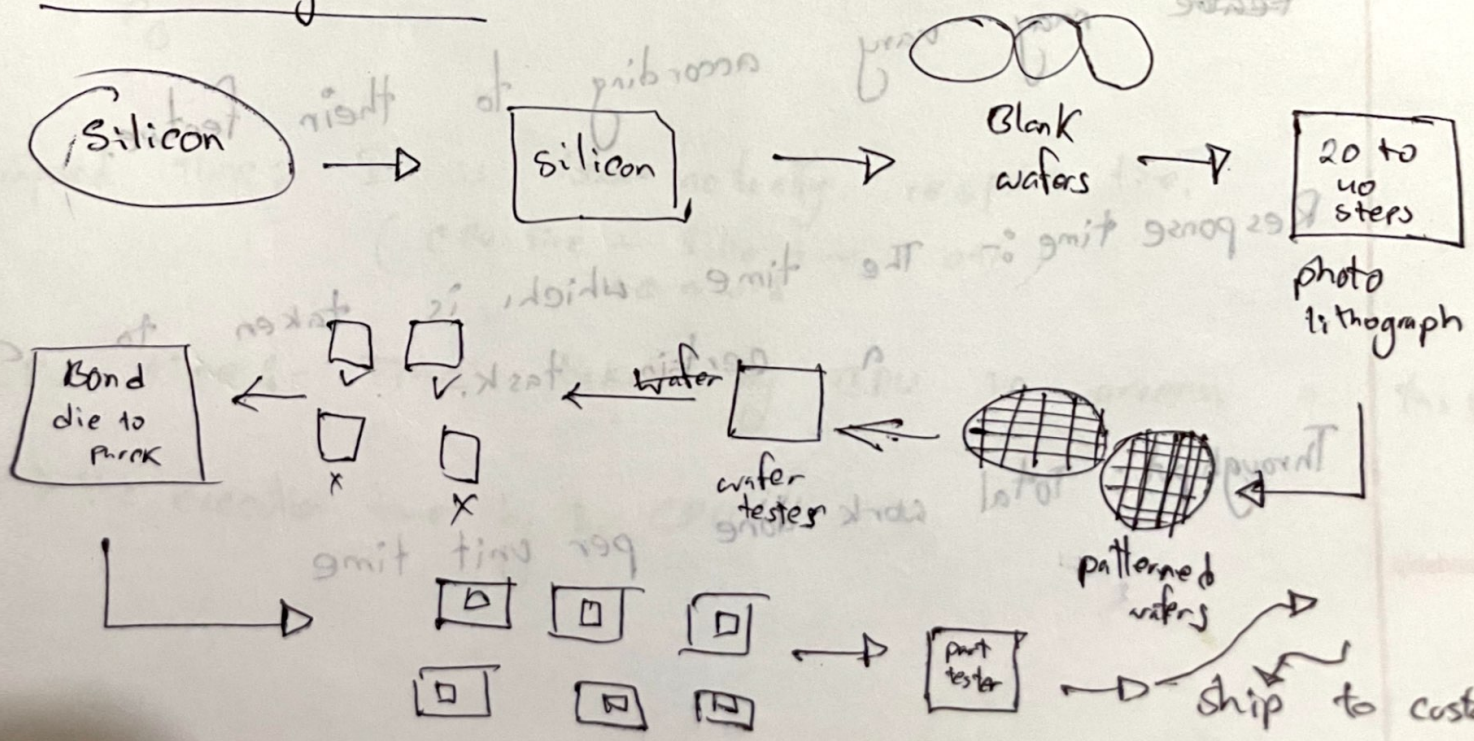
for (i=0, i<=5, i++) { ⇒ Assembly [Loop :

⇒ [00100100 . . .]

ISA

ABI → what type of hardware is needed, it needs the permission.

Manufacturing IC's :-



Subject : Time :

FORMULA:-

(i) Dies per wafer = $\frac{\text{Wafer area}}{\text{Die area}}$

(ii) Cost per Die = $\frac{\text{Cost per wafer}}{\text{Dies per wafer} \times \text{Yield}}$

(iii) Yield = $\frac{1}{\left\{1 + \left(\text{Defects per area} \times \frac{\text{Die area}}{2}\right)\right\}^2}$

Non linear relation to area (three points). [might igiven in theoretical analysation.

☆ Performance ~~Feature~~ may vary according to their feature.

Response time :- The time which is taken to complete a certain task.

Throughput : Total work done per unit time

Scanned with CamScanner

* Time less ≈ better performance.

Performance $= \frac{1}{\text{Execution time}}$

$P_x = n \times P_y$

if in exam :-
X is n times greater than Y then we have to start the calculation using performance

Example :-

Given,

$E_A = 10s$, $E_B = 15s$

$\frac{E_B}{E_A} = \frac{15}{10}$

$\Rightarrow \frac{1/P_B}{1/P_A} = 1.5$

$\Rightarrow \frac{P_A}{P_B} = 1.5$

$\therefore P_A = 1.5 \times P_B$

Elapsed time :- It is ~~stmo~~ actually response time.
(CPU time + Idle time + etc)
+ overhead

CPU time :- Time taken by CPU to process a task

→ The execution time by the CPU

Scanned with CamScanner

Subject : CPU Time ≈ Execution time.

Date :
Time :

* Clock cycle:- 1 high & 1 low is 1 clock cycle

* Clock ~~cycle~~ period:- Time taken to complete 1 clock cycle. /clock cycle time

* Clock rate/clock frequency: The number of cycles per second [Clock cycle time = $\frac{1}{\text{Clock Rate}}$]

CPU Time = CPU clock cycles x clock cycle time

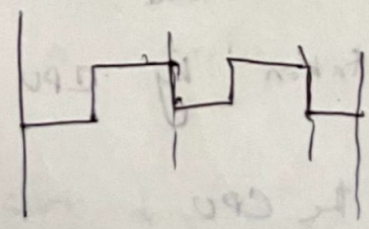
The number of clock cycle in performing a single task = $\frac{\text{CPU. clock cycles}}{\text{clock Rate}}$

=> How much fast ~~must~~ the ~~com~~ PC must be [then we have to work with the clock rate.]

<u>CPI</u> :-

CPU clock cycle = Instruction count x clock cycle per instruction (CPI)

Here,



Subject : Time :

Combining CPI & Intruction count in CPU Time:-

CPU Time = CPU clock cycle × clock cycle time

= Instruction count × CPI × clock cycle time

$$\therefore \text{CPU Time} = \frac{\text{Instruction count} \times \text{CPI}}{\text{Clock } \cancel{\text{cycle time}} \text{ Rate}}$$

For Different Instruction class take different no. cycles

Total Clock cycles (for different classes) = $CPI_A \times \text{Instruction Count}_A + CPI_B \times \text{Instruction Count}_B + \ldots$
$\ldots CPI_n \times \text{Instruction count}_n$

☆ Total clock cycles = $\sum_{i=1}^{n} (CPI_i \times \text{Instruction Count}_i)$

☆ $\therefore$ Weighted Average CPI = $\frac{\text{Total clock cycles}}{\text{Total instruction count}}$

Same ISA = (Instruction Count) same = Same program.

Scanned with CamScanner

Subject : Time :

instructions →

Add	Add
Add	Sub
Add	mul

CPI for add = 2

CPI for add = 2
" " sub = 2
" " mul = 2

For S2:-

Total clock cycle = 1×1 + 2×1 + 3×1
= 9

Total IC = 6

AVG CPI = $\frac{9}{6}$ = 1.5

CPU Time = 5×6×1.5 =

Power Trends

Power = Capacitative load × Voltage^2 × frequency

Subject : Time :

Microarchitecture Enhancement :-

(i) Cache optimization : increase the size & enhancement of cache

(ii) Specialized execution units: GPU for specific tasks.

Software optimization | # Energy efficient computing: turn off the unused component of CPU

Heterogenous computing :- Combine different types of cores. (high performance + energy efficient)

Advanced cooling & packaging

SPEC RATIO

To compare between two CPU performance there is a coorporation "SPEC".

$$\text{SPEC Ratio} = \frac{\text{Reference time}}{\text{Execution time}}$$

$$\text{Geometric mean} = (SR1 + SR2 + SR3 + \dots + SP_n)^{1/n}$$

AMDAHL'S LAW

If we improve 1 task then how the total task ~~we~~ will get improved.

Scanned with CamScanner

Subject : Time :

$T_{improved} = \frac{T_{affected}}{(\text{improvement Factor})} + T_{unaffected}$

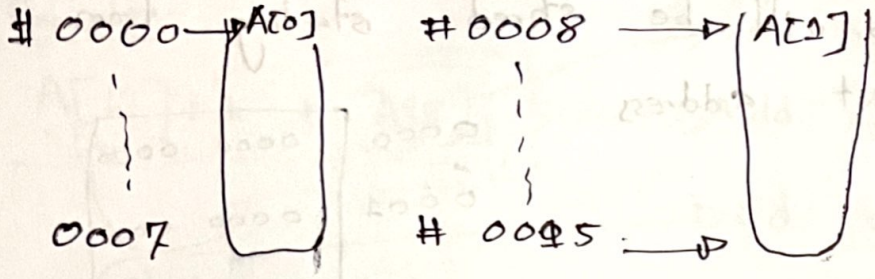
$\downarrow$ n

MIPS

Millions of instructions per second.

$$MIPS = \frac{\text{Instruction Count}}{\text{Execution time} \times 10^6}$$

Subject : Chapter-2 Time :



Arr1 is a Double word sized array.

Arr1 length is 100

Base address of Arr1 = #0000

Find the location of A[2]

Actual address = Base address + offset

$\underbrace{8 \times [\text{index}]}_{\text{offset}}$

1 byte	#	8 bit
1 word		32 "
1 half word		16 "
1 Double word		64 "

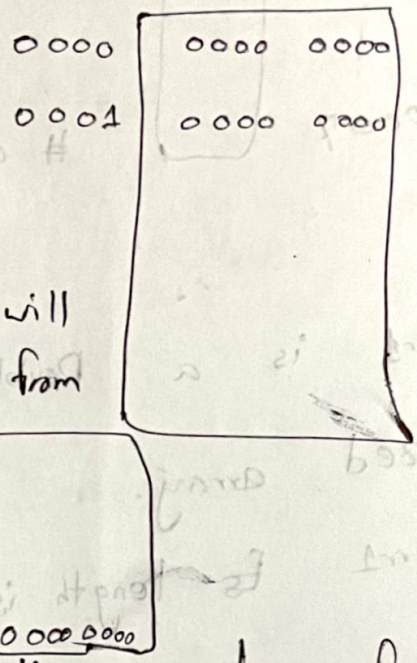
Alignment:- any data will start from multiple of 4

Subject :

Date :

Time :

Big endian :- every 8 bits msb will be stored starting from lowest address



Little endian :- every msb 8 bit will be stored starting from highest address.

why little endian?

→ so that we could perform the operations faster.

Load / store syntax

l [d]

load, data size

ld X_{21}, 16[X_{20}]

stored

destination, source

offset, base register

sd

store, data size

Example :- for double word.

temp reg, index

ld X_5 8 x 8 [X_{22}]

Subject : Risc-V size 64 bit (double word)

Date :
Time :

Example:-

A[12] = h + A[8]

store

ld x_6, 64[x_{22}]

add x_6, x_{21}, x_6

sd x_6, 96[x_{22}]

Add instruction syntax:-

1

$x_5 = x_5 + 2$

2 is in x_7

1st way

Add x_5, x_5, x_7

~~2 is in~~

2nd way:-

Add x_5, x_5, -2 → This line will get executed faster

Signed / Unsigned extensions:-

LB → signed extension

LBU → unsigned "

INSTRUCTION FORMAT:

I type → 2 register, 1 constant

1 source, 1 dest

example:-

ld x_5, 16[x_{21}]

dest, source

S type → 2 register, 1 constant

2 source

sd x_2, 24[x_{20}]

source, source

Scanned with CamScanner

Subject :

$g = h + 2$ $\quad g = x_{20}$

$h = x_{21}$

~~Add~~ Addi $x_5, x_0, 2$ "or" Addi $x_{20}, x_{21}, 2$

Add x_{20}, x_{21}, x_5

extra load

rs → register source

rd → " destination

Add $\underbrace{x_5}_{rd}, \underbrace{x_6}_{rs1}, \underbrace{x_7}_{rs2}$ | R type

~~Add~~ $\underbrace{x_5}_{rd}, \underbrace{x_{10}}_{rs1}, \underbrace{x_5}_{Imm}$ | I type

imm	rs1	func3	rd

ld $\underbrace{x_5}_{rd}, \underbrace{16}_{Imm}[\underbrace{x_{20}}_{rs1}]$

sd $\underbrace{x_{20}}_{rs2}, \underbrace{12}_{Imm}[\underbrace{x_{21}}_{rs1}]$ | S type

properly such that)

CSE340

Chapter-2

Types of instructions

I type :- 2 register, 1 constant (Immediate)

1 source, 1 destination

example :-

Addi, SLLi, SRLi, ORi, ANDi, ld,

Code example :-

ld x5, 16 [x21]

data copy

rd / destination, immediate, source rs1

~~Addi~~ Addi x5, x0, 2

destination, source rs1, constant / immediate

Machine code :

immediate	rs1	func3	rd	opcode
12 bits	5 bits	3 bits	5 bits	7 bits

rd

slli x19, x20, 4

rd, rs1, immediate

REGENT airways
every little thing counts

Scanned with CamScanner

R type:- 3 registers

2 source, 1 destination

example:-

Add, sub, SLL, XOR
OR, AND

Code example:-

Add x_5, x_6, x_7

x_5 → rd (destination)
x_6 → rs1, x_7 → rs2 (source)

Machine Code:

func7	rs2	rs1	func3	rd	opcode
7 bits	5 bits	5 bits	3 bits	5 bits	7 bits

S type:- Store instruction [sd]

2 source, 1 immediate

Code example:- data copy

sd x_{20}, 12 [x_{21}]

x_{20} → source / rs2
12 → immediate
x_{21} → source / rs1

Machine Code:

imm[11:5]	rs2	rs1	func3	imm[4:0]	opcode
7 bits	5 bits	5 bits	3 bits	5 bits	7 bits

rs2 → source
rs1 → base

SB type :- 2 register, 1 branch (Conditional jump)

2 source

example :-

beq, bne, blt, bge

Code example :-

bne X_9, X_{24} exit

rs1 rs2

Machine code :-

imm[12]	imm[10:5]	rs2	rs1	func3	imm[4:1]	imm[11]	opcode
1 bit	6 bit	5 bit	5 bit	3 bit	4 bit	1 bit	7 bit

UJ type :- 1 reg, branch

1 destination

JAL

Code example :-

Jal X_0, 2000

rd

Machine code :-

imm[20]	imm[10:1]	imm[11]	imm[19:12]	rd	opcode
1 bit	10 bit	1 bit	8 bit	5 bit	7 bit

Scanned with CamScanner

U type :- 1 reg , 1 immediate

LUI

rd

(we use it to form 32 bit immediates)

example:-

lui

Syntax:- lui rd, constant

Code example:-

~~lui, rd~~

lui X_{19}, 976

rd constant

Machine Code:-

immediate	rd	opcode
20 bits	5 bits	7 bits

Time :

12 → 0000 00000 1100

11 5 4 0

7 10

what might be the theoretical Q. of SLC1 or SLR1

Shift Operation

	Dec val	Left shift by	
0000 0010	2		2
0000 0100	4	1	2×2^1
0000 1000	8	2	2×2^2
0001 0000	16	3	2×2^3

	Dec val	Right shi by	
0001 0000	16		
0000 1000	8	1	16/2
0000 0100	4	2	16/
0000 0010	2	3	16/

SRL x_{11}, x_{10}, x_9

SRLI x_{11} x_{10} 1

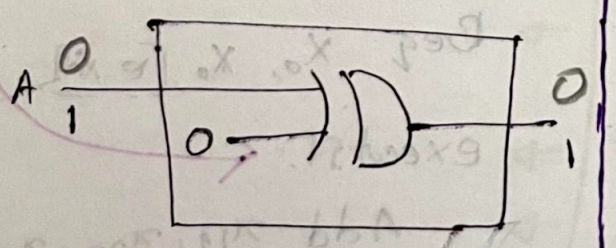
↓ ↓

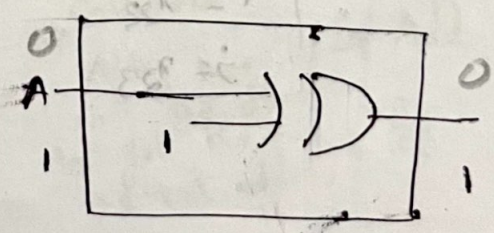
r_d r_{s1}

OR

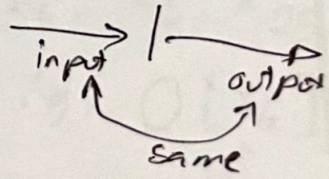
Using OR we add bits in particular position

XOR





Buffer gate



Conclusion :

Subject : Time :

32 bits ← machine code [00 ... 00]

"Every instruction takes 32 bits."

#0000	Line-1
0001	
0002	
0003	
0004	Line-2
0005	
0006	
0007	
0008	Line-3
0009	
0010	
0011	
0012	
0013	
0014	
0015	

32 bits add x_2, x_3, x_5 # Line1

32 bits sub x_2, x_3, x_4 # Line2

32 bits slli $x_2, x_3, 2$ # Line3

(DECISION MAKING)

Normal code to RISC-V code:-

if (i == j):
f = g+h
else:
f = g-h

f = x_{19}
g = x_{20}
h = x_{21}
i = x_{22}
j = x_{23}

→ Beq x_{22}, x_{23}, execute If
→ Sub x_{19}, x_{20}, x_{21}
→ Beq x_0, x_0, exit
→ execut If:-
→ Add x_{19}, x_{20}, x_{21}
→ exit:-

Subject :

Date :

Time :

```
if (i==j):
    f = g+h
i = i+1
```

$f = x_{19}$
$g = x_{20}$
$h = x_{21}$
$i = x_{22}$
$j = x_{23}$

Beq

```
Beq  x22, x23, L1
Beq  x0, x0, exit
L1:
    Add  x19, x20, x21
exit:
    Addi x22, x22, 1
```

Bne

```
Bne  x22, x23, exit
Add  x19, x20, x21
exit:
    Addi x22, x22, 1
```

Loop:-

```
while (save[i]==k):
    i = i+1
    a = a+1
```

$i = x_{22}$
$k = x_{24}$
$a = x_{23}$
$\underbrace{\text{save base}}_{\text{Double word array}} = x_{25}$

save[0] → 0x8 → 0
save[2] → 2×8
save[i] → index(i) × 2^3

```
Loop:
    ld   x9, 0[x25]
    Bne  x9, x24, break
    Addi x22, x22, 1
    Add  x23, x23, 1
    Beq  x0, x0, Loop
break:
```

This wont work for all cases as it is static

```
Loop:
    SSli  x5, x22, 3
    Add   x6, x5, x25
    ld    x9, 0[x6]
```

offset calculation

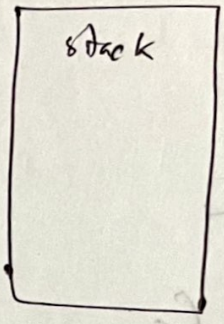
Adding base with the offset to get the actual address

actual address

Subject : Time :

→ What is the main difference between same reg & temp reg

<u>Static data</u> :- constant, fixed length data.



← what are storing in every segment of this box.

<u>Frame pointer</u>: helps to mange stack better

handles → Saved reg, arg reg, return add reg

→ If there are more than 8 parameters we will use frame pointer.

---

want to store

$x_{15} = 15$

→ Addi $X_{15}, X_0, 15$

LUI → using this we can use it as 32 bit immediate reg.

<u>EXAMPLE:-</u>
(Using LUI)

Save 23304 $\xrightarrow{\text{binary}}$ 1011 011 0000 1000

MSB → 1011 ; LSB → 1000

(we have to use LUI as it exceeds 12 bit)

Subject :

Date :

Time :

64 bit ⇒

0000 0000 0000 0000 0000 0000 0000 0000 0000

31

0000 0000 0000 0101 1011 0000 1000

5 12 11 0 LSB

2824

→

0000 - - - - 0000 0000 0000 0000 0000 0101 0000 0000 0000

1011 0000 1000

Addi x_{15}, x_{15}, 2824

U type instruction

→ LUI

SB type instruction :

offset → no. of lines × $\frac{\text{place bit}}{\text{9 bit}}$

12 → 0000 0000 110~~0~~ = 12

0000 0000 $\underbrace{110}_{6 \times 2}$ = 12

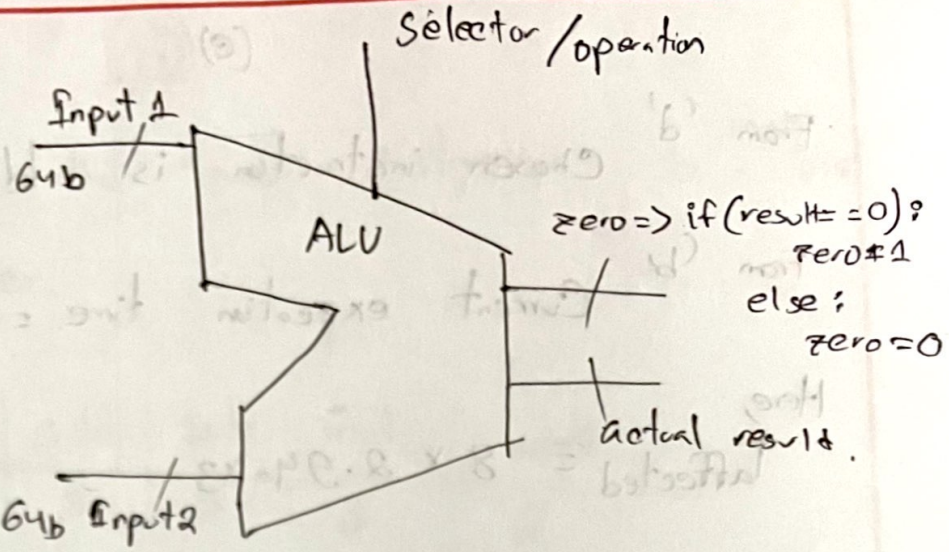
→ actual offset

Scanned with CamScanner

Subject :

Chapter - 3

```
   1000
   1001
  ------
   1000
  00000
 000000
 000000
 ------
 001000
```

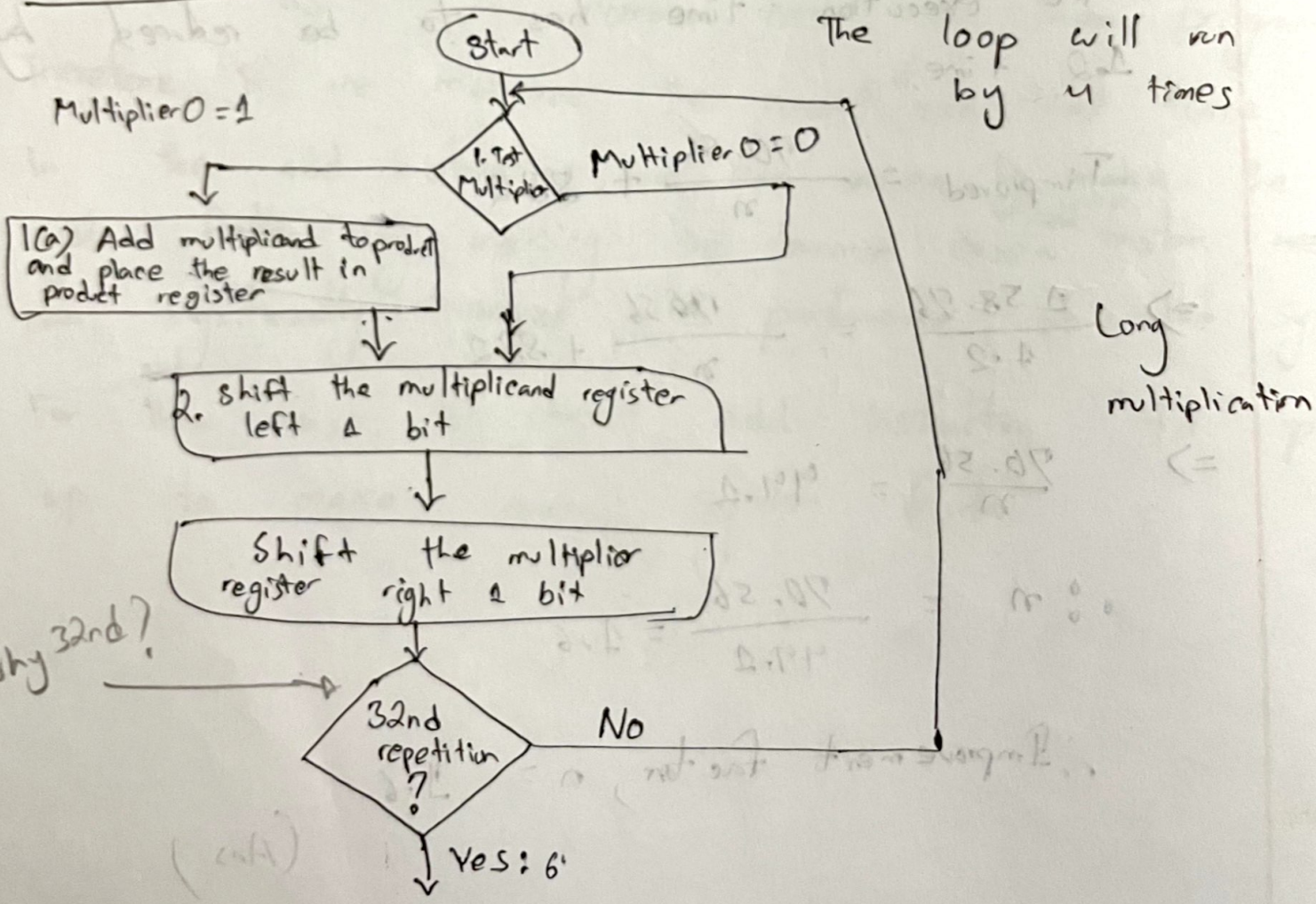


Multiplicand → [00 . . . . 00] [actual multiplicand 32 bit]

32 bit

Only zeros

LONG MULTIPLICATION



Subject :

Multiplicand × Multiplier

1000 × 1001 ← Here is 4 bit that means the loop will run by 4 times

Date :
Time :

iteration	Multiplicand	Multiplier	Product
0	0000 1000	1001	0000 0000
1	0000 1000	1001	0000 1000
	0001 0000	~~0100~~ 1001	0000 1000
	0001 0000	0100	0000 1000
2	0010 0000	0100	0000 1000
	0010 0000	0010	0000 1000
3	0100 0000	0010	0000 1000
	0100 0000	0001	0000 1000
4	0100 0000	0001	0100 1000
	1000 0000	0001	0100 1000
	1000 0000	0000	0100 1000

← 1/2 marks in exam

No. of bits in multiplier
= No. of iteration

Algorithm

```
if (iteration <= multiplier bit length):
    if (multiplier0 == 1):
        product_MSB = Multiplicand + product_MSB
        product = right shift product by 1
    elif (multiplier0 == 0):
        product = right shif " " "
```

Subject : $\underbrace{1000}_{\text{multiplicand}} \times \underbrace{1001}_{\text{multiplier}}$

Time :

Optimized multiplication

Iteration	Multiplicand	Product (product MSB)
0	1000	0000 1001 (multiplier)
1	1000	1000 1001 0100 0100
2	1000	0010 0010
3	1000	0001 0001
4	1000	1001 0001 0100 1000

Characteristics of a Normalized number :-

$\underbrace{___}_{\text{should be 1 digit to the left of decimal point and it should be non zero}} . ___$

(i) 123.456 [not in normalized mode more than 1 digit]

(ii) 1.236 [normalized mode]

(iii) 0.123_{10} [not in normalized]

(iv) 9.123 [normalized mode]

Scanned with CamScanner

Subject : Chapter - 3 - (ii)

Date :
Time :

Biased Ex = (Bias + Actual Exp) | Bias = $2^{n-1} - 1$

S	Ex	Frac
1	7	15

→ Biased Ex = (63 + (-1))
= 62

Bias = $2^7 - 1$
= 63

~~Ex~~ Biased exponent range:-

$0 - 2^n - 1$

$= 0 - 2^7 - 1$

$= 0 - 127$

$= 1 - 126$

(10.5423)10 store using Single ~~Simple~~ precision.

Example :-

Convert 50.6749 to 32 bit IEEE / single precision.

50 = 110010

.6749 = 10 1011 0011

32	16	8	4	2	1
1	1			1	
			1	1	0

Now,

50.6749 = 110010 . 101011 0011 × 2^0

= (+)1 . 10010 101011 0011 × 2^5 — Actual exponent

fraction

(+)ve sign

Bias = $2^{8-1} - 1$

why 7? → 7

$= 2^7 - 1$

$= 127$

Single precision 8 bit

Biased exponent = (127 + 5)

= 132

= $(1000\ 0100)_2$

Double precision 11 bit

0	1000 0100	10010101011.0000 0000 000
Sign	Biased Exp	fraction

Biased exp. Range:-

$= (0 - 2^n - 1)$

$= (0 - 2^8 - 1) = (0 - 255)$
$= (0 - 254)$

[(1 - 127) to (254 - 127)]

Actual exp range.

Friendship

Scanned with CamScanner

Subject : PROCEDURE CALLING (JAL & JALR)

Date :
Time :

JAL → JAL x_1, procedureLabel

fixed

stores the address to which line it will come back in "x_1"

Any instruction size = 32 bit

cale(a,b,c): [callee]

main ( ) [caller]

1 (0)

2 (4) (1)

x_1, cale(1,2,3) 3 (8)

4 (2) we returning to the next line (12)

JALR → JALR x_0, $0[x_1]$

why x_0? we don't need to know which line we would return in main function.

we know the physical address which line we will return.

To store 1 reg in a stack we need 8 locations.

Register = 64 bit

How To normalize a number:-

EXAMPLE:-

$(1234.5678)_{10}$

$= 1.2345678 \times 10^3$

$\rightarrow 11011.0012$

$= 1.1011001 \times 2^4$

MORE DIGITS TO LEFT

(i) Have to ~~#~~ Left shift the decimal point.

(ii) The no. of times we are left shifting ~~we~~ will be converted to exponent. (added)

EXAMPLE:-

$= (0.0000101)_2$

$= 1.01 \times 2^{-5}$

~~MORE DI~~ zero to ~~right~~ left to decimal point

(i) Right shift the decimal point.

(ii) The no. of times we are ~~to~~ right. shifting will be converted to exponent (subtracted)

IEEE FLOATING POINT FORMAT:-

	sign bit	Exponent	fraction	
Single Precision	1 bit	8 bit	23 bits	32 bit
Double Precision	1 bit	11 bits	52 bits	64 bit

sign bit = 0 => (+)ve

" " = 1 => (−)ve

Biased exponent range always unsigned.

~~Conv~~ Conversion of signed number to unsigned number is bias

single Precision binary range: 0 to 2^8-1

= 0 to 255

= 1 to 254

Double Precision binary range:

0 to $2^{11}-1$

= 0 to 2047

= 1 to 2046

Scanned with CamScanner

Subject : custom

1	6	8
Sb	Exp	frac

Time :

☆ In IEEE ex floating point formal what do we understand by the exponent?

→ It actually helps us to understand that how smaller or larger number we could store

☆ Bias $= 2^{n-1} - 1$ | n = size of exp field

☆ Biased exponet = Actual exponent + Bias

$2^{8-1} - 1$
$= 2^7 - 1$
$= 127$

Q Given, a number IEEE754, 25 bit floating representation find the actal decimal number. Where size of the fraction field is 10 bits. A = 0X0800025

Solution:-

0 8 0 0 2 5

0000 1000 0000 0000 1101 0010 0101

×

000	0	1000 0000 0000 11	0100100101
	sign bit	Biased exp	fraction

Subject : Time :

$(1000000000011) = (8195)_{10}$

Actual exponent $= 8195 - (2^{14} - 1)$

$= 8195 - 8191$

$= 4$

$(.010010101) = .2861$

$\Rightarrow (-1)^0 \times (1 + .2861) \times 2^4$

$\Rightarrow 1.2861 \times 2^4$

Subject :

The numbers are stored in memory locations then we to

The memory locations are stored in registers x_{10}, x_{11} then we have load to a register.

Double Pre.

flw f_1, $0(x_{10})$

Double Pre.

flw f_2, $0(x_{11})$

faddd f_3, f_1, f_2

fsw f_3, $0(x_{13})$

Handout - (IV)
Chapter - 3
(Surprise Quiz)

Range = $2^n - 1$

[$\because n$ = no. of bits]

Bias = $2^{n-1} - 1$

Biased exponent = Actual exponent + Bias.

MORE FRACTION BIT = MORE ACCURACY

Scanned with CamScanner

Subject :

Date :

Time :

PC

#8000

8001	XXXX XXX0
8002	0101 1010
8003	0XXX 0011
8004	0XXX XXXX
⋮	
8009	

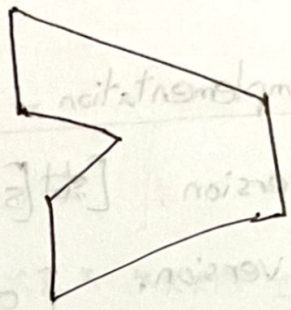
instruction memory

XXXX XXX0 0101 1010 0XXX 0011

funct7 rs2 rs1 funct3 rd

0XXX XXXX

opcode



Topics:-

(i) Instruction fetch

(ii) " Decode

(iii) Reg file read

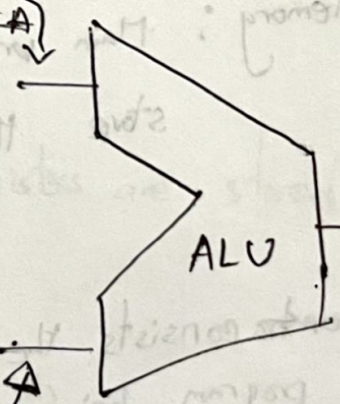
(iv) ALU

(v) Data Memory access

(vi) Write Back

X_{12} = 0x1234AB789ABCDEF01

~~Actual A~~



It is the whole ~~ist~~ instruction itself

what is it?

32 bit

Immediate generation unit

64 bit

take input instruction not the hex val.

then it will extract te 12 bit. Then it will expand 12 bit to 64 bit.

Actual address = Base + Offset

1010 1100 1110 1111

1110 1110 1100 1100

Subject : Chapter-4 (in a processor) Time :

To know about the CPU time we have to know the Instruction Count (IC). To find out the IC we have to know about the ISA or the language we are using for the program (RISC-V).

RISC-V implementation - 2 types:-

(i) Simplified version. [Slow]

(ii) Pipelined version. [optimized]

INSTRUCTION EXECUTION STEP:-

#Instruction Memory :- Holds the instructions only

#Data Memory : Main memory. In case of load/store this memory gets accesed.

PC:- Program Counter register

The instruction which we are currently executing and the address of that instruction are stored in PC

STEPS :-

(i) PC ~~contra~~ consists the address of the instruction in the program being executed.

Using this address from PC we fetch the instruction from the instruction memory.

Subject :

Date :

Time :

(ii) The fetched instruction gets decoded. Then the registers ~~which~~ associated with the decoded instruction will become activated so that it ~~an~~ can ~~readed~~ read data/write data.

~~By d~~ By decoding we try figure out the source of the registers

Every instruction size is 32 bit and every cell of the IM is 8 bit. So 4 cell required to store 1 instruction

That's why: PC+4

(iii) Example :-

PC = #6000

step-1

Instruction Memory

#6000	xxxx	xxx 0
#6001	0101	1010
#6002	0xx1	0011
#6003	0xxx	xxxx

step 2

xxxx xxx0 0101

0xxx 0011 0xxx xxxx

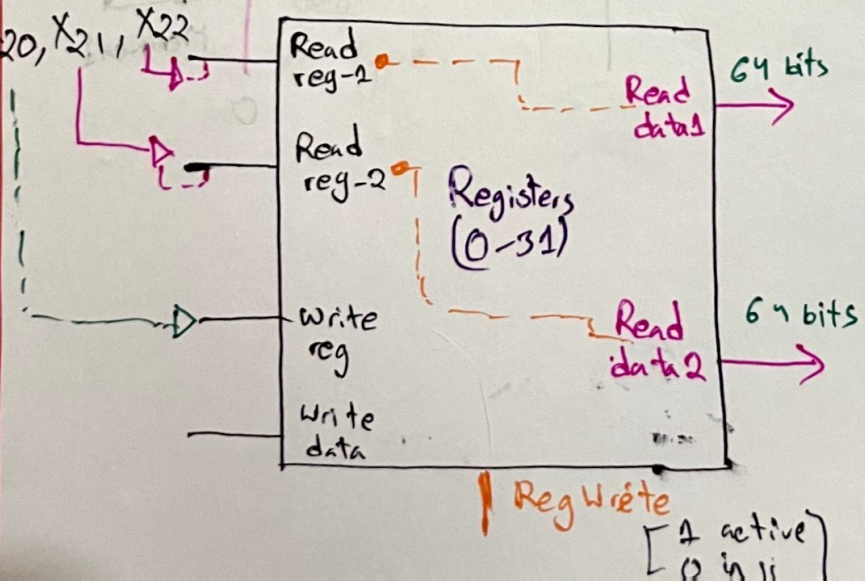
Decodes step-3

REGISTER FILE:

In a register file the 32 registers are stored.

→

add X20, X21, X22



Friendship

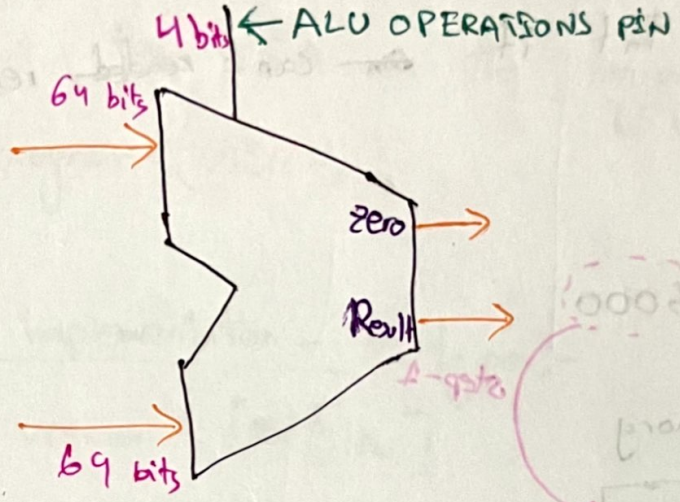
Scanned with CamScanner

Subject :

Time :

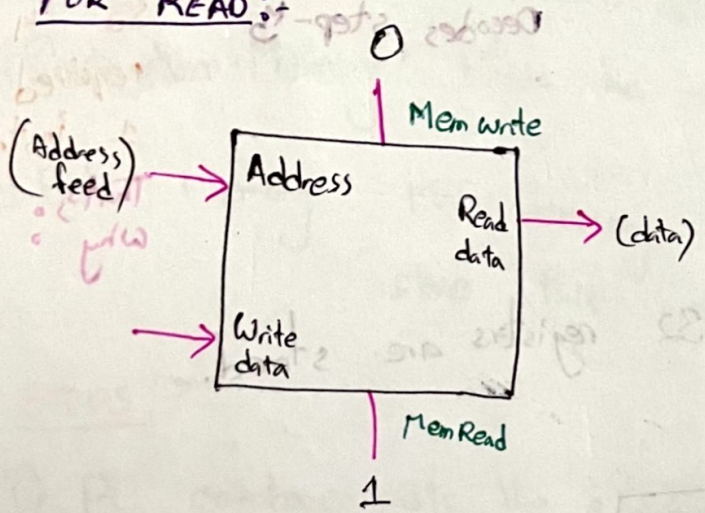
Arithmetic Logic Unit (ALU)

ALU tells that if the result is 0 or not

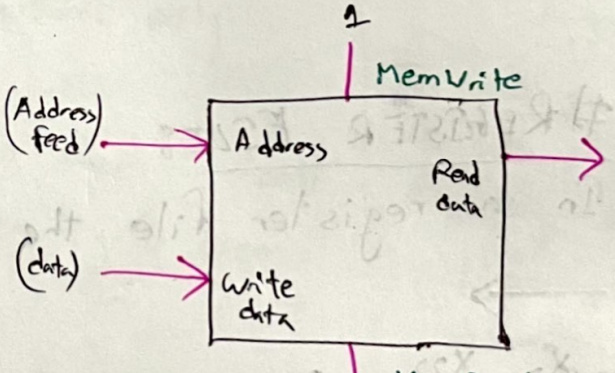


DATA MEMORY UNIT :-

FOR READ :-



FOR WRITE :-



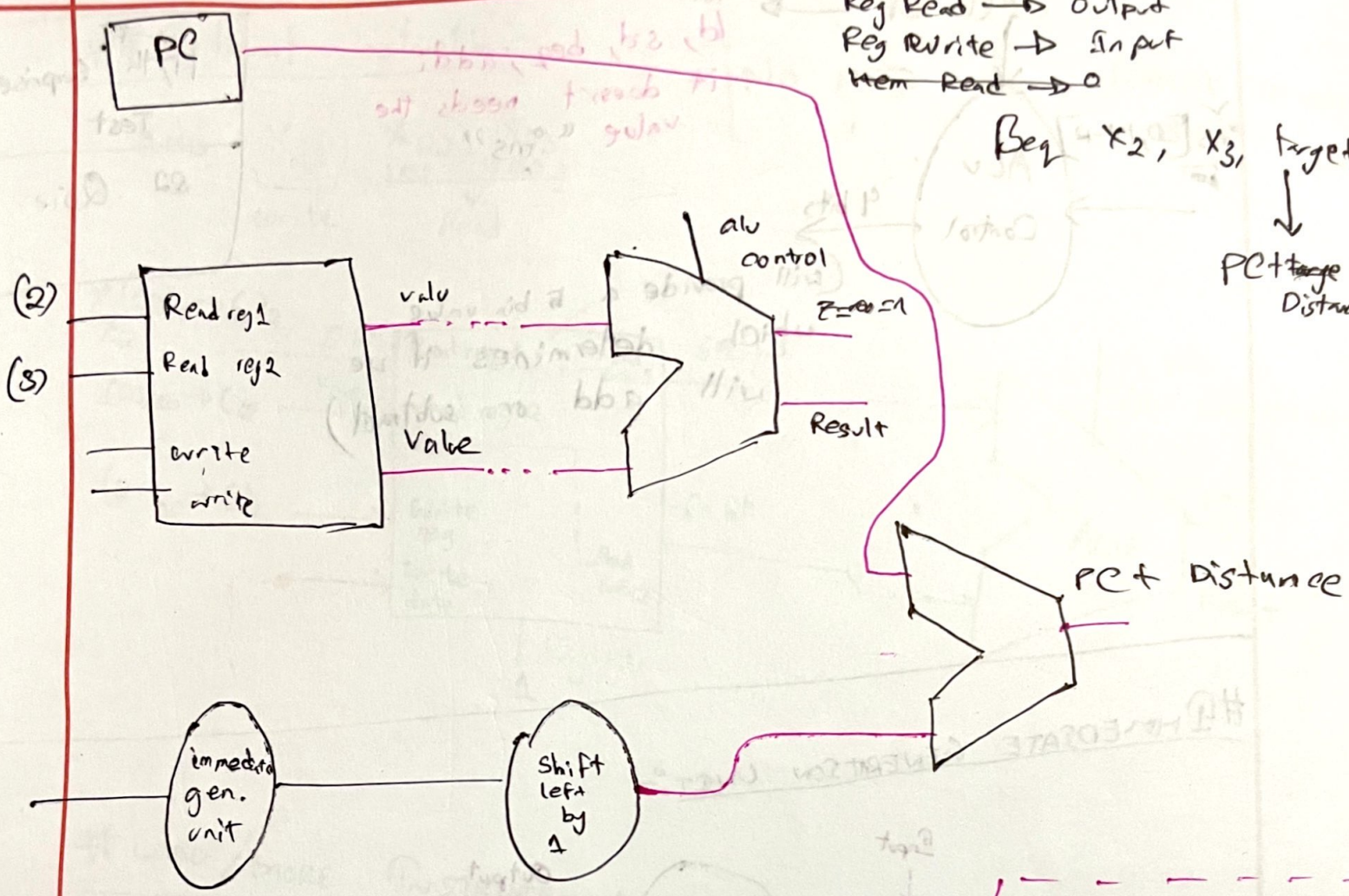
Subject :

Date : 10/09/24
Time :

Mem Write → Input } Data Memory

Reg Read → output
Reg Write → Input
~~Mem Read~~ → 0

Beq x_2, x_3, target
↓
PC + ~~Target~~ Distance



Q. type
what is the value generated in different pin of control unit

~~What~~ ALU OP ⇒

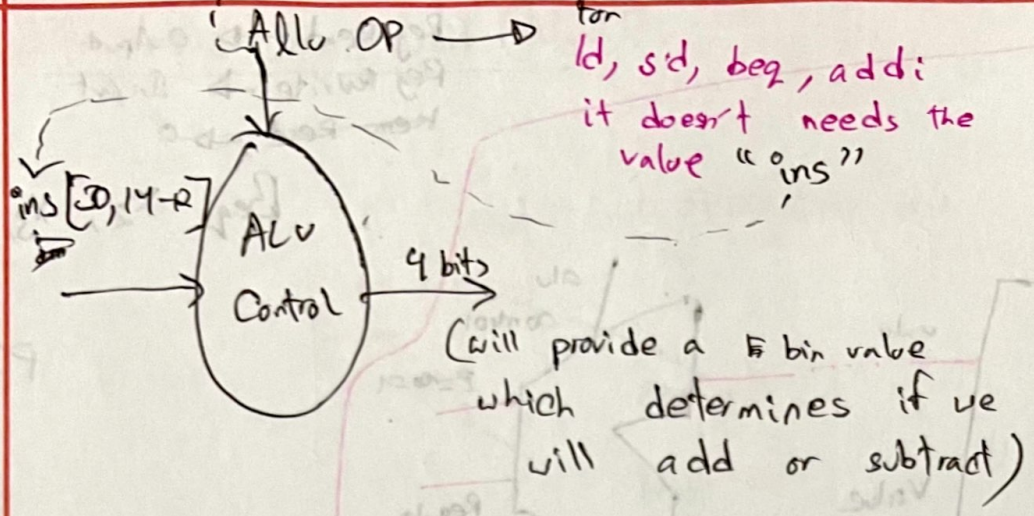
00 – load / store / addi
01 – Beq
10 – R type

Subject : Control unit

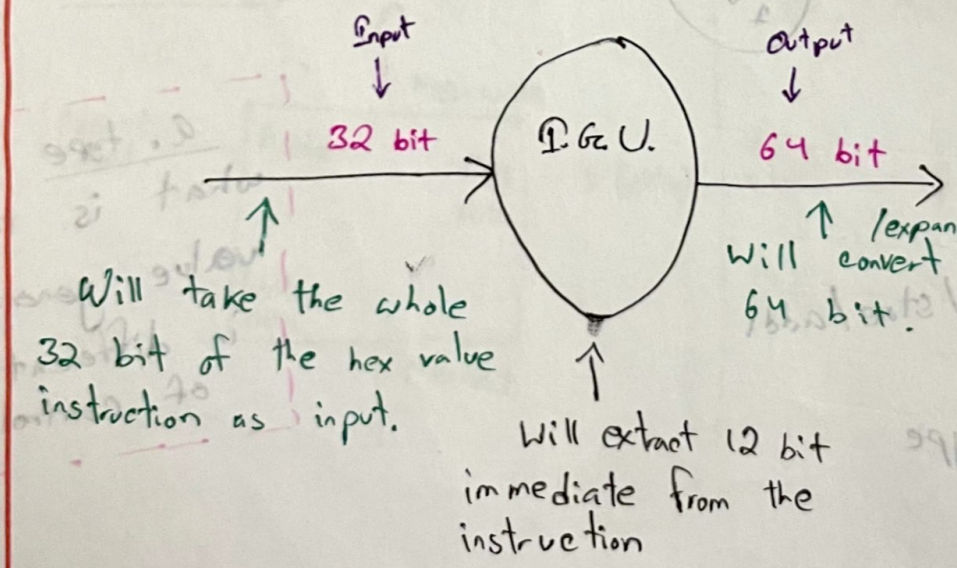
Date :
Time :

17th Surprise Test

Q2 . Quiz



IMMEDIATE GENERATION UNIT :-

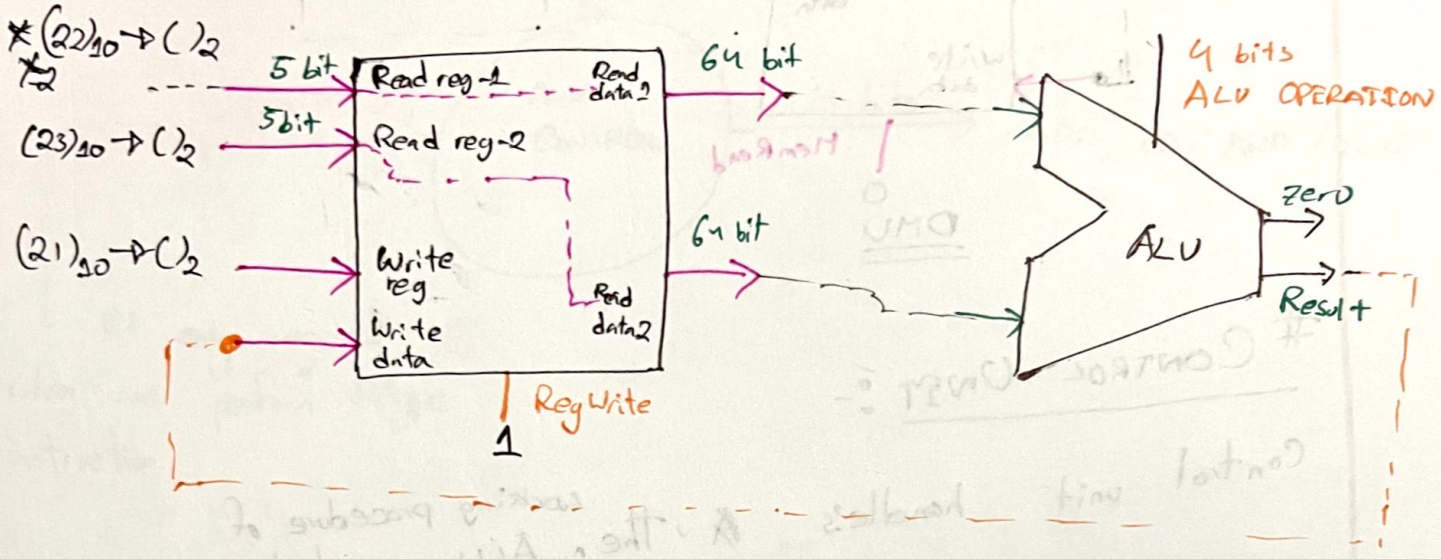


R-format instruction :-

[NO NEED OF DATA MEMORY UNIT]

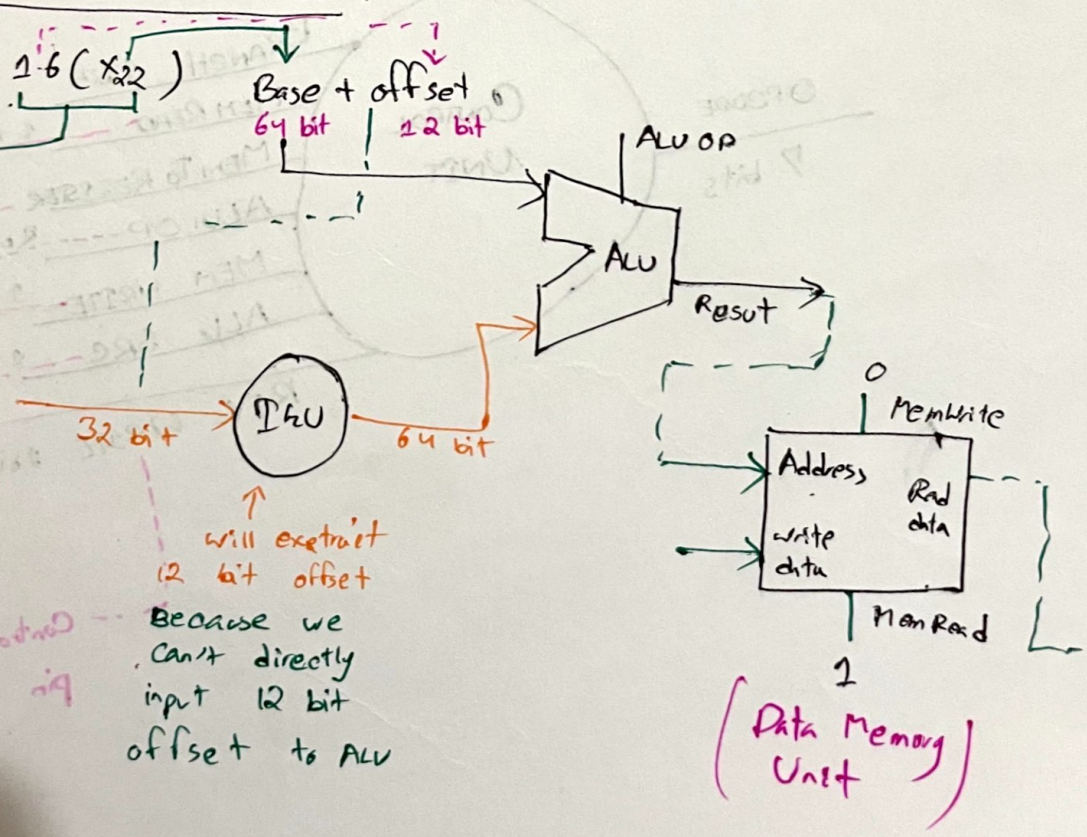
add X_{21}, X_{22}, X_{23}

write — Read (rs1, rs2)



LOAD / STORE INSTRUCTION :

LD X_{21}, 16(X_{22})



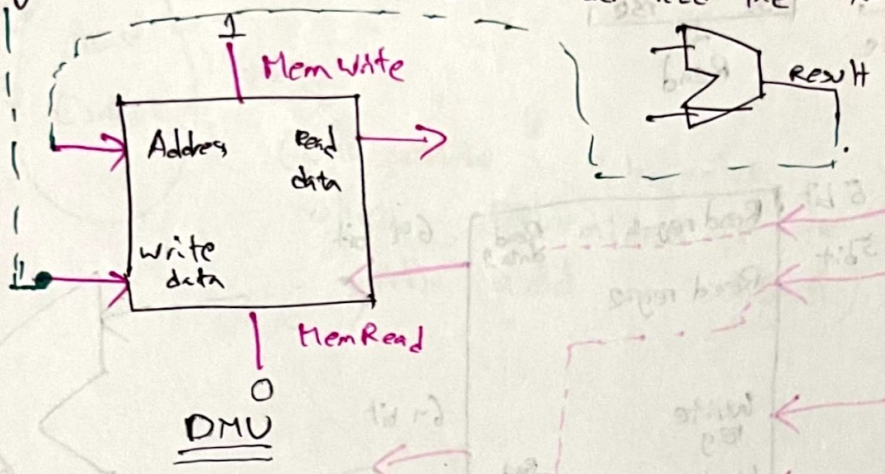
Subject :

Date :

Time :

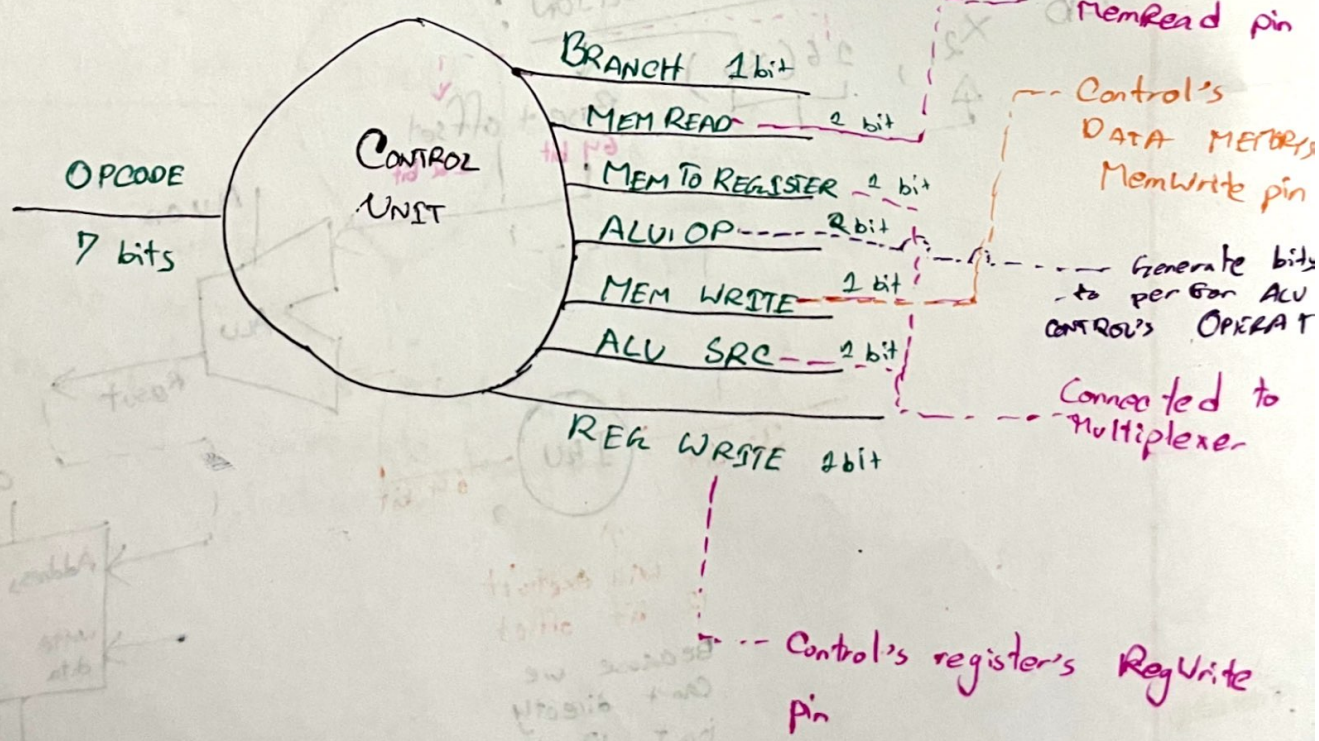
SD X_{21}, 16 (X_{23}).

(everything is same as LD just ~~add~~ feed the Write data to DMU)



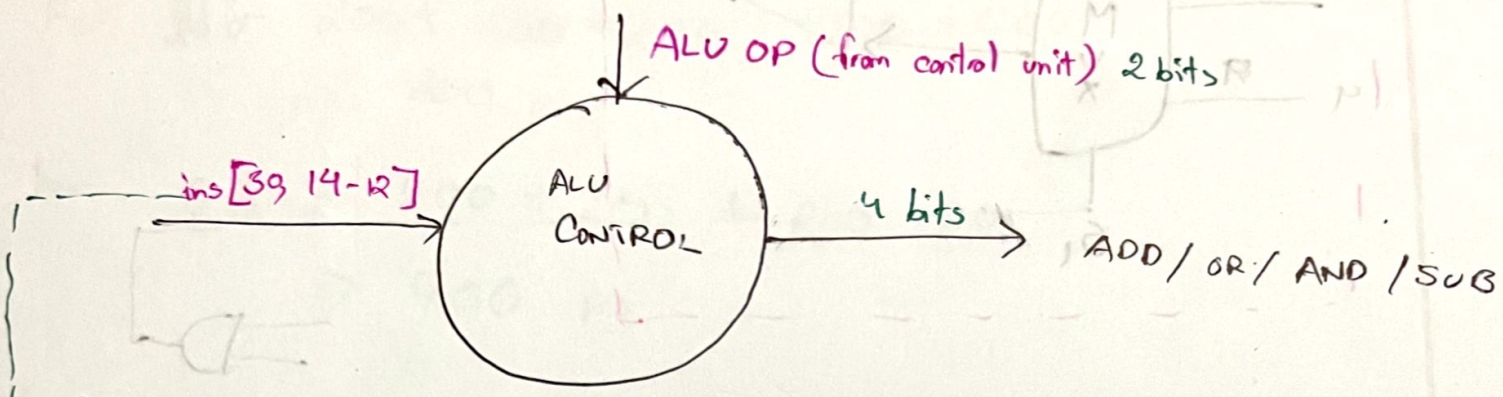
CONTROL UNIT :-

Control unit handle's ~~A~~ working procedure of the ALU control :

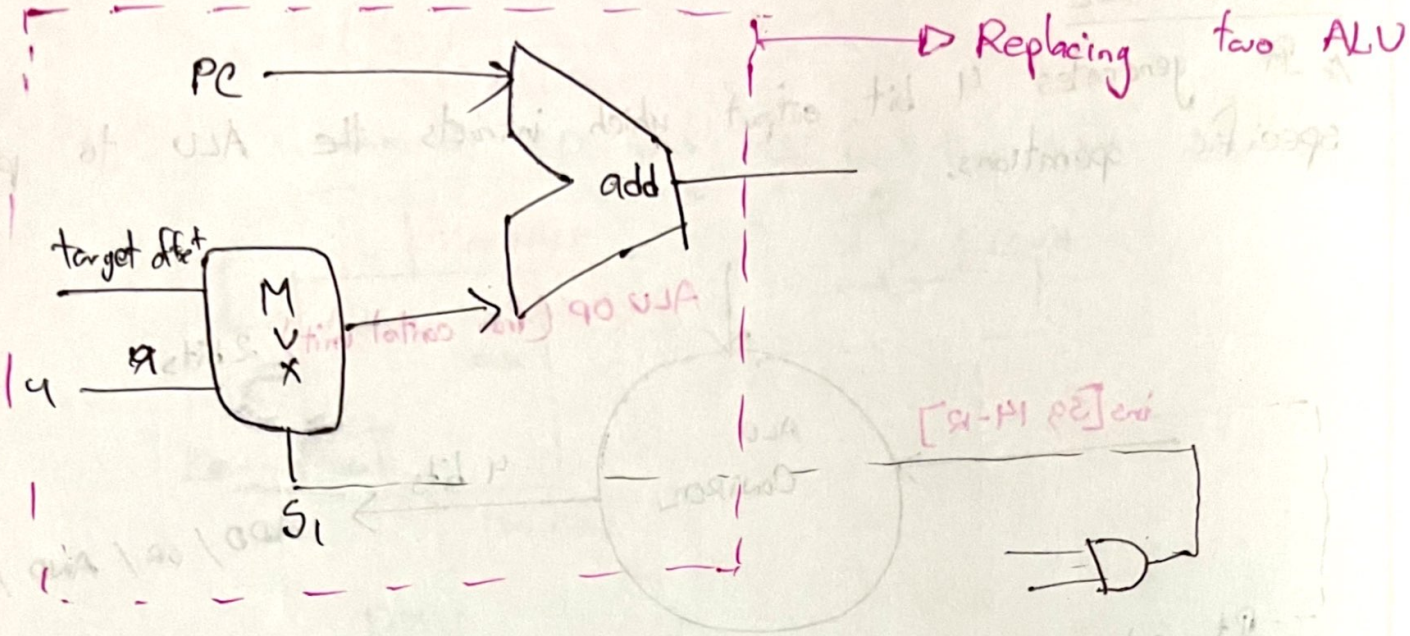


ALU CONTROL

It generates 4 bit output which intructs the ALU to perform specific operations.



It only requires when we perform R-type instruction.



☆ Mux ~~instuction~~ input reversed → R type & I type. Then what will be the signal of ALU src.

☆ Draw the data path for

Add x_2, x_3, x_4
Addi x_3, x_3, x_4

} only → draw the selected pins.

☆ Might be question ~~which~~ where we will ~~m~~ replace a component from the data path. ~~It~~ The q.1 will ask for if the data path or not.

Scanned with CamScanner

PIPELINE :-

Performance :-

For ld :- 200 + 100 + 200 + 200 + 100

=> 800 ps

For sd :- 200 + 100 + 200 + 200

=> 700 ps

For R-format :- 200 + 100 + 200 + 100

=> 600 ps

For Beq :- 200 + 100 + 200

=> 500 ps

For single ~~cycly~~ cycle what will be the clock period ?

→ 800 ps [Longest delay determines the clock period]

Clock period = 800 ps

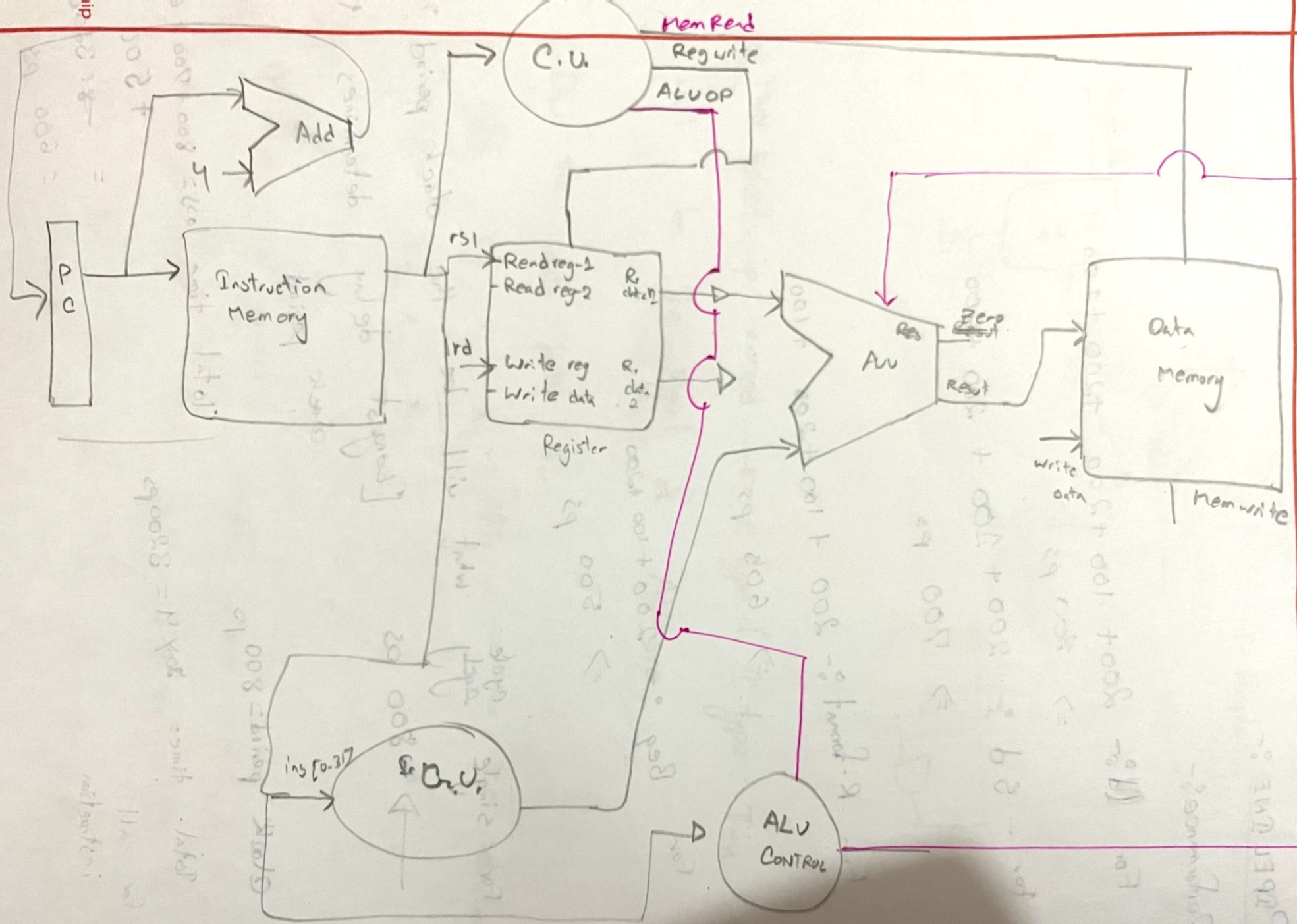
Total time = 800 × 4 = 3200 ps for all instruction

Total time loss = 800 + 700 + 600 + 500

= ~~8~~ 3200

= 600 ps

Scanned with CamScanner



Subject :

Date :

Time :

Scanned with CamScanner

Answer to the q. no. - 01

Multiplicand = 13 = 1101

Multiplier = 17 = 10001

No. of iteration = 5

Iteration	Multiplicand	Product
0	01101	00000 10001
1	01101	01101 10001 00110 11000
2	01101	00011 01100
3	01101	00001 10110
4	01101	00000 11011
5	01101	01101 11011 00110 11101

Result = 00110 11101

Answer to the q. no.- 02

Given,

$$0 \times ABB9609 = \underbrace{1}_{\text{sign bit}} \,|\, \underbrace{0101\ 011}_{\text{Biased exponent}} \,|\, \underbrace{1011\ 1001\ 0110\ 0000\ 1001}_{\text{fraction field}}$$

Bias exponent bits = 7

$$\text{Bias} = 2^{7-1} - 1$$
$$= 2^6 - 1$$
$$= 63$$

Biased exponent = 010 1011
= 43

$\therefore$ Actual Exponent = 43 − 63
= −20

$\therefore$ Sign bit = 1

$$\text{fraction field} = 0.\underset{-1}{1}\underset{-3}{0}\underset{-4}{1}\ \underset{-5}{1}\underset{-8}{0}\underset{-9}{1}\ \underset{-10}{0}\underset{-11}{1}\underset{-12}{0}\ \underset{}{0}\underset{}{0}\underset{}{0}\underset{}{0}\ \underset{-17}{1}\underset{}{0}\underset{}{0}\underset{-20}{1}$$

$$= 0.7241297$$

Now,

$$\text{Decimal value} = (-1)^{1} \times (1 + 0.7241297) \times 2^{-20}$$
$$= -1.7241297 \times 2^{-20}$$
$$= -1.644258 \times 10^{-6}$$

(Ans)

Scanned with CamScanner

Time :

Answer to the q. no. - 03

Given,

70.78955

$\longrightarrow$

70 = 1000 110

.78955 = 1100 1010

$\therefore$ 70.78955 = 1000 110. 1100 1010

= 1. 0001 1011 0010 10×2^{6} (A.E)

.78955 × 2 = 1.579

.579 × 2 = 1.158

.158 × 2 = 0.316

0.316 × 2 = 0.632

0.632 × 2 = 1.264

.264 × 2 = 0.528

0.528 × 2 = 1.056

0.056 × 2 = 0.112

Biased exponent bit = 9

Here,

Bias = $2^{9-1} - 1$

= $2^{8} - 1$

= 255

Sign bit = 0

Biased exponent = 255 + 6

= 261

= 1000 0010 1

fraction field = 36 - (9+1) = 26

Now,

0	1000 0010 1	0001 1011 0010 1000 0000 0000 00
sign bit	B.E.	F. field.

= 0 × 41.466 A.000

(Ans)

Subject : Time :

Answer to the q. no.- 04

(a)

Given,

$50.7869 + 79.83 - 29.58$

Here,

$50 = 110010$

$.7869 = .11001001$

$\rightarrow$

$50.7869 = 110010.11001001$

$= 1.10010110\ 01001 \times 2^5$

$.7869 \times 2 = 1.574$
$.574 \times 2 = 1.148$
$.148 \times 2 = 0.296$
$.296 \times 2 = 0.592$
$0.592 \times 2 = 1.184$
$.184 \times 2 = 0.368$
$.368 \times 2 = 0.736$
$0.736 \times 2 = 1.472$

and,

$79 = 1001111$

$.83 = 1101\ 0100$

$\rightarrow$

$79.83 = 1001111.1101\ 0100$

$= 1.0011\ 1111\ 0101\ 00 \times 2^6$

$.83 \times 2 = 1.66$
$.66 \times 2 = 1.32$
$.32 \times 2 = 0.64$
$.64 \times 2 = 1.28$
$.28 \times 2 = 0.56$
$0.56 \times 2 = 1.12$
$0.12 \times 2 = 0.24$
$0.24 \times 2 = 0.48$

Scanned with CamScanner

and,

$29 = 11101$

$.58 = 1001\ 0100$

$29.58 = 11101.\ 1001\ 0100$

$= 1.\ 1101\ 1001\ 0100 \times 2^4$

$.58 \times 2 = 1.16$

$.16 \times 2 = 0.32$

$.32 \times 2 = 0.64$

$.64 \times 2 = 1.28$

$.28 \times 2 = 0.56$

$.56 \times 2 = 1.12$

$.12 \times 2 = 0.24$

$0.24 \times 2 = 0.48$

Now,

$50.7869 + 79.83$

$= 1.1001\ 0110\ 01001 \times 2^5 + 1.0011\ 1111\ 0101 \times 2^6$

$= 0.1100\ 1011\ 0010\ 01 \times 2^6 + 1.0011\ 1111\ 0101 \times 2^6$

$= (0.1100\ 1011\ 0010\ 01 + 1.0011\ 1111\ 0101) \times 2^6$

$= 10.0000\ 1010\ 0111\ 01 \times 2^6$

$= 1.0000\ 0101\ 0011\ 101 \times 2^7$

$$
\begin{array}{r}
1.0011\ 1111\ 0101 \\
0.1100\ 1011\ 0010\ 01 \\
\hline
10.0000\ 1010\ 0111\ 01
\end{array}
$$

Subject:

Again,

(Sum) Δ - 29.58

$= 1.0000\ 0101\ 0011\ 101 \times 2^7 - 6\ 1101\ 1001\ 01 \times 2^7$

$= 1.0000\ 0101\ 0011\ 101 \times 2^7 - 0.0011\ 1011\ 0010\ 1 \times 2^7$

$= (1.0000\ 0101\ 0011\ 101 - 0.0011\ 1011\ 0010) \times 2^7$

$= 0.1100\ 1010\ 0001\ 101 \times 2^7$

$= 1.1001\ 0100\ 0011\ 01 \times 2^6$

(Ans)

Scanned with CamScanner

(b)

Given,

64.2486 * 49.1832

Here,

64 = 1000 000

.2486 = .0011 1111

64.2486 = 1000 000. 0011 1111

= 1. 0000 0000 1111 11 $\times 2^6$

.2486 × 2 = 0.497
0.497 × 2 = 0.994
.994 × 2 = 1.988
.988 × 2 = 1.976
.976 × 2 = 1.952
.952 × 2 = 1.904
.904 × 2 = 1.808
.808 × 2 = 1.616

and,

49 = 1100 01

.1832 = 0010 1110

49.1832 = 1100 01. 0010 1110

= 1. 1000 1001 0111 $\times 2^5$

.1832 × 2 = 0.366
.366 × 2 = 0.732
.732 × 2 = 1.464
.464 × 2 = 0.928
.928 × 2 = 1.856
.856 × 2 = 1.712
.712 × 2 = 1.424
.424 × 2 = 0.848

Now,

$= (1.0000\ 0000\ 1111\ 11 \times 2^6$
$* 1.1000.1001\ 0111\ 0 \times 2^5)$

$= (1.0000\ 0000\ 1111\ 11 * 1.1000\ 1001\ 0111\ 0)\ 2^{6+5}$

$= 1.1000\ 1010\ 1111\ \cancel{0001}\ 0011 \times 2^{11}$

(Ans)

Subject :

Answer to the q. no. - 05

Given,

$28.4810 - (-4.0210)$

$= 28.4810 + 4.0210$

Here,

$28 = 1110\ 0$

$0.4810 = .0111\ 1011$

$28.4810 = 11100.0111\ 1011$

$= 1.1100\ 0111.1011 \times 2^4$

$0.4810 \times 2 = 0.962$

$.962 \times 2 = \cancel{1.9624}\ 1.924$

$.924 \times 2 = 1.848$

$.848 \times 2 = 1.696$

$.696 \times 2 = 1.392$

$.392 \times 2 = 0.784$

$.784 \times 2 = 1.568$

$0.568 \times 2 = 1.136$

and,

$4 = 100$

$.0210 = 0000\ 0101$

$4.0210 = 100.\ 0000\ 0101$

$= 1.0000\ 0001\ 01 \times 2^2$

$.0210 \times 2 = 0.042$

$0.042 \times 2 = 0.084$

$0.084 \times 2 = 0.168$

$.168 \times 2 = 0.336$

$.336 \times 2 = 0.672$

$.672 \times 2 = 1.344$

$.344 \times 2 = 0.688$

$.688 \times 2 = 1.376$

Scanned with CamScanner

Subject

Time :

Now,

$$1.1100\ 0111\ 1011 \times 2^4 + 1.0000\ 0001\ 01 \times 2^2$$

$$= 1.1100\ 0111\ 1011 \times 2^4 + 0.\cancel{0010\ 0000\ 0010\ 1}\ 0100\ 0000\ 0101 \times 2^4$$

$$= (1.1100\ 0111\ 1011 + 0.\cancel{0010\ 0000\ 0010\ 1}\ 0100\ 0000\ 0101) \times 2^4$$

$$= 10.\cancel{1110\ 0111\ 1101}\ 0000\ 1000\ 0000 \times 2^4$$

$$= 1.0000\ 0100\ 0000 \times 2^5$$ (Ans)

Again,

$$\text{Bias} = 2^{8-1} - 1 = 2^7 - 1 = 127$$

$$\text{Biased exponent} = \cancel{4}\ 5 + 127 = \cancel{132}\ \cancel{131} = 132$$

$$\text{Range} = 0 \text{ to } 2^8 - 1 = 0 \text{ to } 255 = 1 \text{ to } 254$$

Here,

$$1 \leq 132 \quad \text{and} \quad 132 \leq 254$$

$\therefore$ No overflow or underflow.

Subject :

Time :

Answer to the q. no. - 06

Given,

$$0.0001\ 01 \times 2^{-85} * 10.1 \times 2^{-90}$$

$$= 1.01 \times 2^{-85-4} * 1.01 \times 2^{-90+1}$$

$$= 1.01 \times 2^{-89} * 1.01 \times 2^{-89}$$

$$= (1.01 * 1.01) \times 2^{-89-89}$$

$$= 1.1001 \times 2^{-178}$$

Here,

Bias exponent bits $= 18 - (12+1) = 5$ bit

Bias exponent $= 15 - 178 = -163$

Range $= 0$ to $2^5 - 1$

$= 0$ to 31

$= 1$ to 30

$\therefore -163 < 1$

There will be underflow.

endship

Scanned with CamScanner

Subject :

Date :

Time :

Answer to the q. no. - 07

Here,

$g \leftarrow x_{10}$, $h \leftarrow x_{11}$, $i \leftarrow x_{12}$, $j \leftarrow x_{13}$

CODE :-

```
flw  f1 , 0[x10]
flw  f2 , 0[x11]
flw  f3 , 0[x12]
flw  f4 , 0[x13]
fadd.s  f5, f1, f2
fadd.s  f6, f3, f4
fsub.s  f7, f5, f6
fsw  f7 , 0[x14]
```

Friendship

Scanned with CamScanner

Subject :

Answer to the q. no. - 08

(a)

Bias is added to the actual exponent in the floating point representation in order to convert signed number to unsigned number. As a result, the hardware can treat the exponent as a positive integer. It will ~~th~~ simplify the operations.

When bias is added to the ~~bias~~ positive exponent the exponent value becomes greater than the bias. Again, when bias is added to the negative exponent the sign becomes positive and ~~to~~ the value becomes less than the bias.

Scanned with CamScanner

Subject :

Date :

Time :

(b)

In optimized multiplication there is no requirement ~~for~~ of any extra register which is used in traditional long multiplication. Also, the multiplicand is added to the product msb which is not for long multiplication. We have to perform only right shift to product and add multiplicand when required in optimized multiplication. So, it reduces the number of operation in optimized multiplication. Therefore, the efficiency and performance gets improved in optimized multiplication than compared to long multiplication.

Scanned with CamScanner

Subject : Single cycle Datapath

1 instruction = 1 cycle.

Date :
Time :

PIPELINE :-

Different executions of instruction ~~gets~~ are overlapped. [Parallelism occurs]

Different instructions are executed at the same time.

At a single clock cycle, we run/execute 1 instruction

NOTE :- Execution of instructions into several stages, allowing multiple instructions to proceed simultaneously

EXAMPLE :-

Instruction fetch = 10 Ps

Register file read/write = 5 Ps

Memory read / write = 6 ps

ALU Operation = 10 ps

Add . . . → IF + Reg R. + ALU OP + 10 + 5 + 10 + 5 = 3

SD . . . → IF + Reg R. + ALU OP + M. 10 + 5 + 10 + 6 = 31

LD . . . → IF + R.R. + ALU OP + Mem R + Reg

= 10 + 5 + 10 + 6 + 5

= 36 ps

CLOCK PERIOD OF PIPELINED → largest delay of a stage

Highest time for instruction = 36 ps

Clock period = 36 ps

✓ CLOCK PERIOD OF SYSTEM :- longest delay to run an instruction

✓ Clock Cycles

✓ Total time to run a instruction sequence = clock period x instruction

✓ single cycle means = 1 instruction per clock cycle.

✓ No of Clock cycle = stage + (n-1) [For Piplined]

n = no. of instruction

Subject :

Date :
Time :

STRUCTURAL HAZARD:-

If at the same clock cycle we are using a single instruction step/Process ~~many times~~ more than 1 times

NOTE:-
At a single clock cycle we cannot use a single clock cycle more than 1 time

★ Why we have seperate memory for instruction storing & data storing?

→ In order to avoid structural hazard we have divided our memory into two different categories.
- → Instruction memory
- → Data "

DATA HAZARD:-

When try to decode or read data from a certain register before its data gets updated then its ~~da~~ ~~completion~~ ~~of upda~~ data H hazard. (Because, then we are just reading a garbage value from that register)

Ways to solve data hazard:-

(i) STALLING

(ii) FORWARDING

(iii) CODE SCHEDULING

There is no data hazard in Lare

SD X_{22}, 10[X_{21}]

Add X_{22}, X_{22}, X_{21}

Because, the ~~Read~~ data isn't getting updat

Subject :

Date :

Time :

Scanned with CamScanner