

Chapter - 2

Instructions, Language of the Computer

MIPS Architecture 32 bit

ম্যামোরি দেখা এবং bit length 32 bit

Computer এর main memory রে প্রতিক ডেটা store করায়।
 ২১৮১ এর main memory রে প্রতি 32 bit এর data
 store করা, 32 bit এর উচ্চকাষ্টের data main memory
 রে store করা হয়।

32 bit data are called word.
 known as memory.

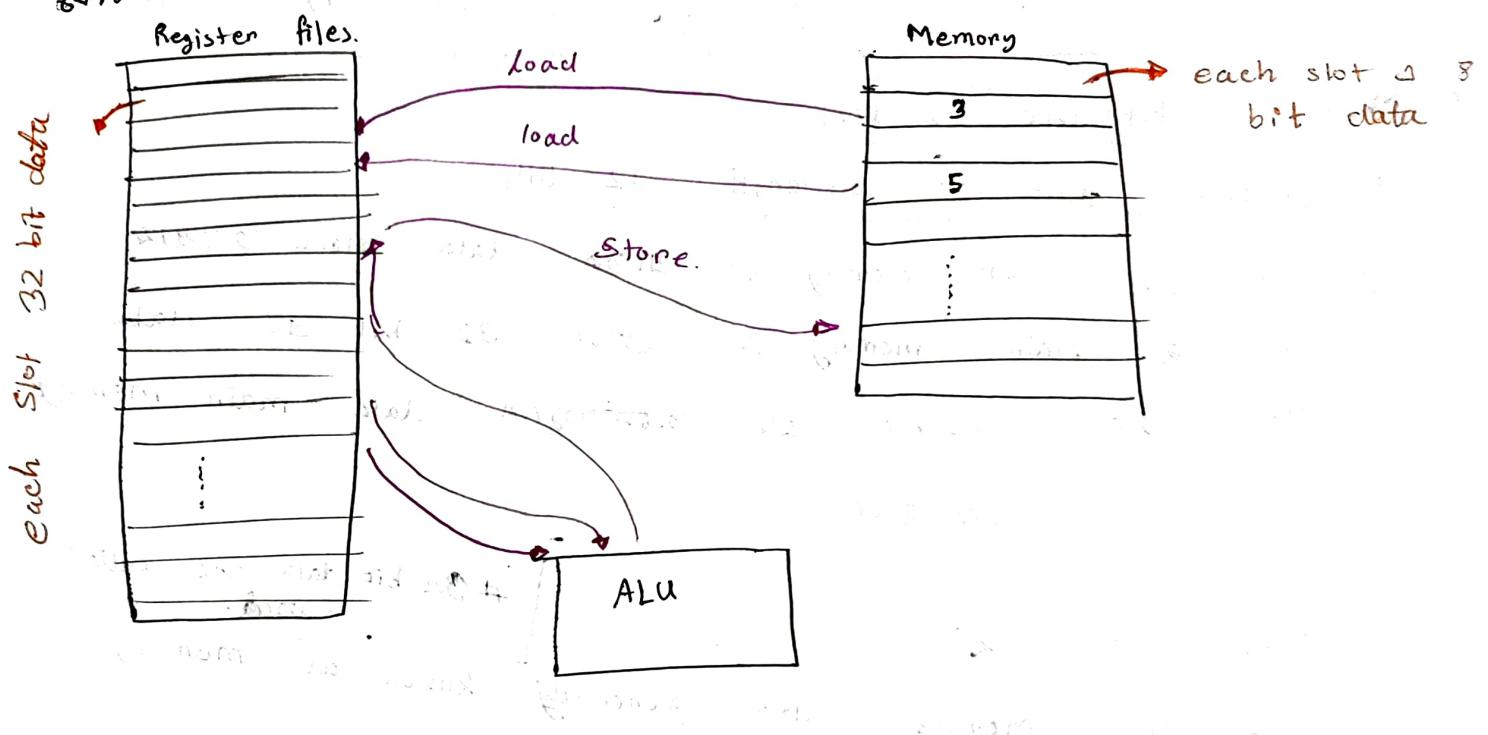
Memory Unit রেটি।

⇒ Main Memory also generally known as memory.
 ⇒ register file.
 32 টি register রেই একই memory।
 each register 32 bit hold করে।

main memory রে একের memory slot রয়েছে, তারের
 data memory রে বাধা রাখিবলৈ কিন্তু register file
 এর স্থান লাম্বা কাণ্ডা।
 limited data রাখা স্থান রয়েছে, কাণ্ডা reg. file
 data রাখার স্থান রয়েছে ২১৭ ৩২ টি slot রয়েছে।

CPU register file রয়েছে। use করে স্থান বেঁধে।
 data তাক �frequently use করা লাগে।

স্কোর অরিথমেটিক ওপেরেশন



for example $3 + 5$

memory 6200 3 5 reg file এ নিয়ে যাবো,
then 3200 6200 data ALU এ যাবো তখন
ALU calculated result এ আবৰণ কৰবো

then reg file দেবো result
এ পেতে পেতে ফিরো তখন reg file দেবো result

১) memory দে এন্টি স্টোর কৰতে

destination source
add (a, b, c) source 2
b + c result : a : to store 262

$5 \rightarrow$ 00000101 (memory to)
 32 bit C reg file \hookrightarrow 32 bit \hookrightarrow convert করো)

একটি memory slot এর address টা 5 bit
 represent করো 32 bit লাগে, কারণ $2^5 = 32$.

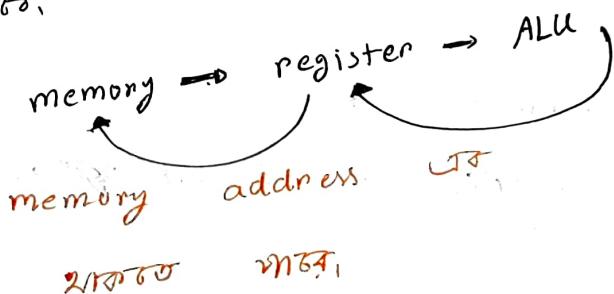
Memory 64MB data load করি তার store করি.

Load = retrieve করি, memory 64MB ফিল.

read operation

store = write operation

ALU (Arithmetic Logic Unit), register 64MB data access



32 combination

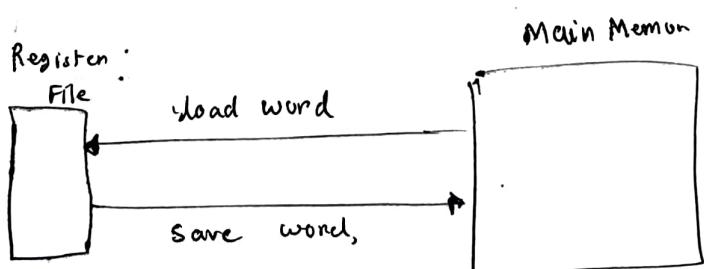
00000
00001
00010
00100
00011
⋮
11110
11111

Memory Operands:

- First data from main memory to ALU then result to reg. file
- Register file is a medium to store frequently used data.
- reg. file then medium to store main memory data in reg. file
- data use ALU Then result to Instruction perform Arithmetic operation
- Main memory is used for composite data (Arrays, structures, dynamic data, stack)

- Memory is byte addressable. Address identifies an 8 bits.
- Words are aligned in memory.
- Words must be a multiple of 4. \checkmark
- Address must be a multiple of 4. \checkmark

8 bit = 1 byte.

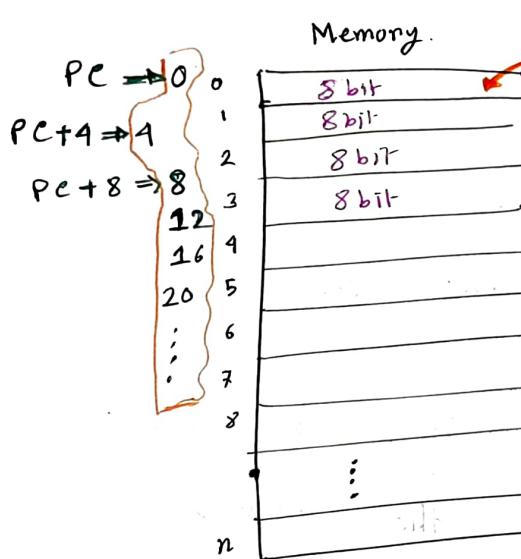
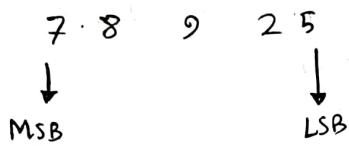


MIPS is Big Endian.

\Rightarrow MSB ~~not~~ byte at least address of a word.

\Rightarrow Little Endian: LSB at least address

LSB higher address
↑ store.



8 bit of data = 1 byte data

1 slot = 8 bit data

4 slot = (4×8) = 32 bit data

8 bit = 1 byte
32 bit = 4 byte
= 1 word.

* যদি মেমোরি একটি architecture হলে, 32 bits কর্তৃত একটি slot করে 4 টি 8-bit দিয়ে প্রক্রিয়া করা হবে।

memory slot নাম।

* প্রথম initial PC (Program point) কর্তৃত একটি slot করে 32 bit দিয়ে প্রক্রিয়া করা হবে।

1st data occupy করতে memory করা হবে।
(0-3)

2nd data occupy করতে memory করা হবে।
(4-7)

Memory എം അടിസ്ഥാന സൗ എഡ്രസ് n ബിറ്റ് സ്റ്റേരീസ്
represent കുറച്ചെണ്ണം 2^n മാറ്റു എഡ്രസ് കമ്പിഷൻ 21720 മീറ്റർ

- Memory എം അടിസ്ഥാന സൗ എഡ്രസ് 7 ബിറ്റ് represent കുറച്ചെണ്ണം
 2^7 മാറ്റു എഡ്രസ് കമ്പിഷൻ 21720 മീറ്റർ + location 5
Still 8 bit data hold മാറ്റു.

32 bit arch pe 64 bit increment കുറഞ്ഞ

64 bit arch pe 8 കുറഞ്ഞ increment കുറഞ്ഞ

■ If you address a memory slot by using 7 bits. Determine the size of the memory?

$$\Rightarrow \text{memory locations} = 2^7 \\ \therefore \text{size} = (2^7 \times 8) \text{ bits}$$

∴ MIPS architecture

address combination

follows this so, 2^{32} bits

location 21720

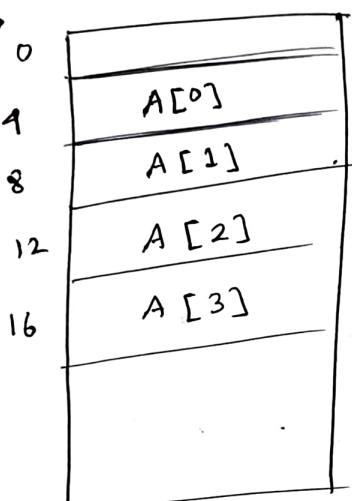
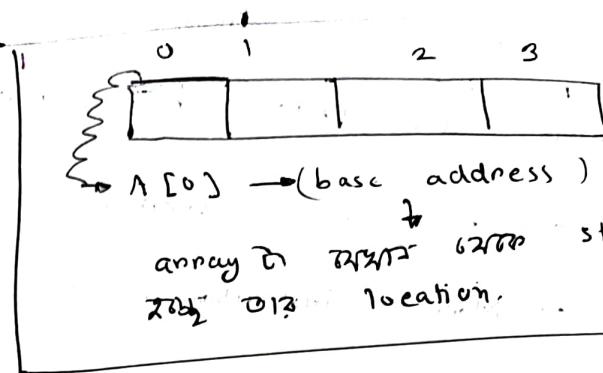
Memory address কোন করা:

$$g.t = h + A[3]$$

Given that,

initial address = 4

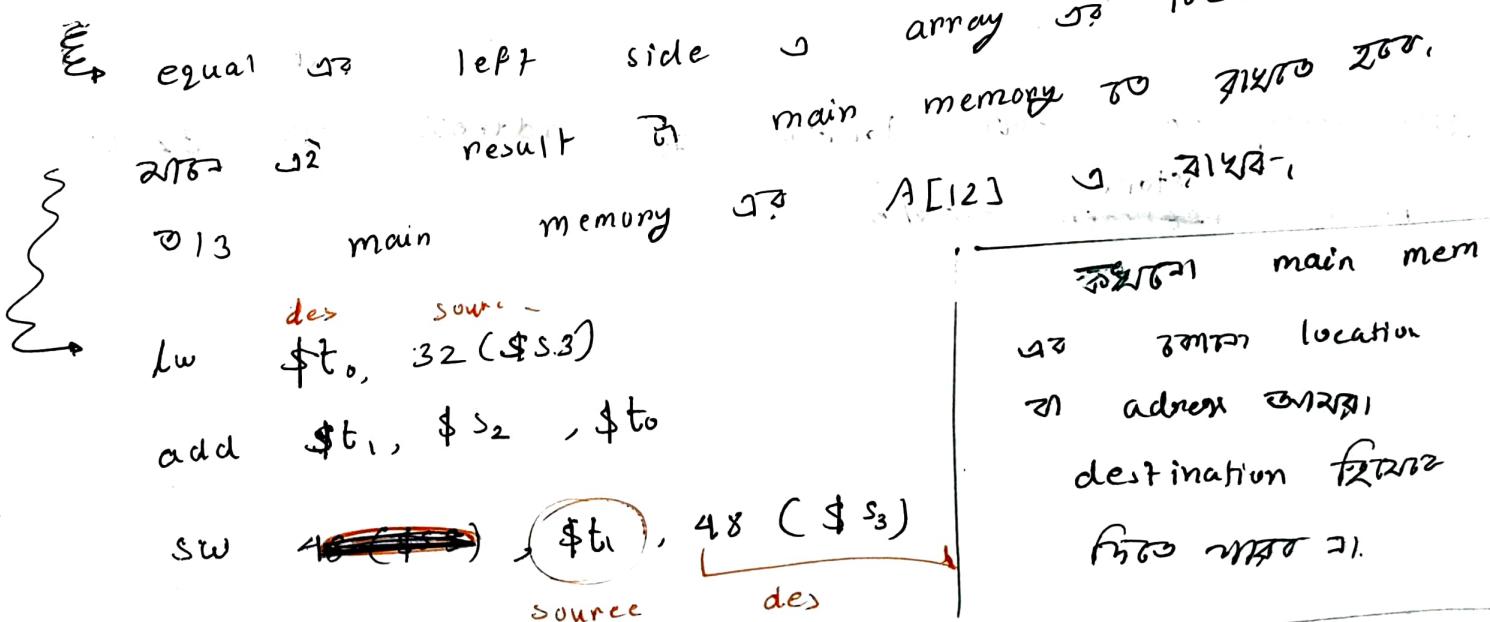
Find the address of $A[3]$.



ক্ষেত্র level
ক্ষেত্র address
 $\therefore A[3] \text{ কর্তৃত } 264$
 $= (3 \times 4) = 12$

address = initial address + 12
= $4 + 12$
= 16

$A[12] = h + A[8];$
base address up to A in \$53



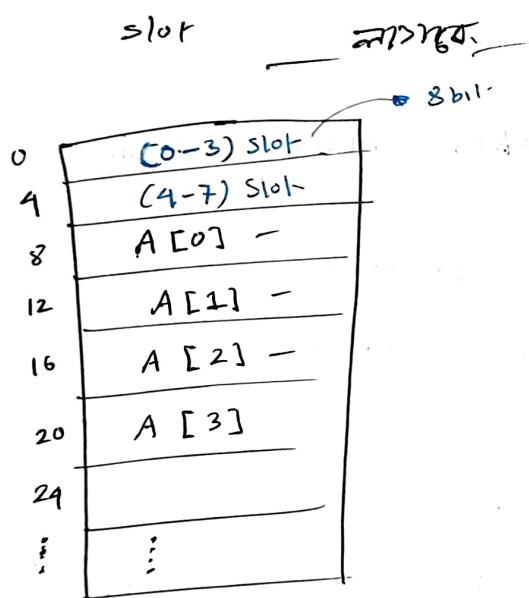
Register vs Memory

- Registers are faster than memory.
- Operating on memory data requires load and store.
- Compiler uses registers for variable as much as possible.
 - More instruction to be executed
- Compiler uses registers for less frequently used variables.
 - Only spill to memory
- ⇒ Registers optimization is important

MIPS Arch can support up to 32 address line
So an address could be 0x AB CD E 1 2 3 in
hexadecimal
8 hex digit;
 $1 \text{ hex} = 4 \text{ bit}$
 $\therefore 8 \text{ hex digit} = 32 \text{ bits}$

- Get memory to get location to address. 32 bit
- Store represent $\frac{1}{2^32}$

- register file \hookrightarrow 32 bit register slot \Rightarrow 1 word = 32 bit = 1 byte data 216bit.
- memory \hookrightarrow 32 bit slot \hookrightarrow 8 bit 216bit, 32 bit MIPS
- Architecture follow 32 bit, but 32 bit. So, memory \hookrightarrow 32 bit slot 8 bit
- data 32 bit \Rightarrow memory \hookrightarrow 32 bit slot 8 bit
- Therefore, memory \hookrightarrow 32 bit slot 4 bit



We have to retrieve the data of $A[3]$. To do that at first we need to find the ~~destination~~ address of $A[3]$.

\Rightarrow From the figure
base address of the array = 8 to slot

$A[0]$ \hookrightarrow address 6400 32 data skip 2160
2160, 1 word 32 data \hookrightarrow 32 4 slot
2160.

\therefore Number of slot to be skipped = (3×4)
 $= 12$ slots.

$$\begin{aligned}
 & \text{of } A[3] \\
 \therefore \text{Destination address} &= \text{Base Address} + \text{number of slots to be skipped} \\
 &= 8 + 12 \\
 &= 20
 \end{aligned}$$

~~✓~~ Formula:

MIPS or 32 bit (or 32)

$$\text{address} = (4 \times \text{index number}) + \text{base address}$$

64 bit architecture (or 32),

$$\text{address} = (8 \times \text{index number}) + \text{base address}$$

Arithmetic and logical operation takes place in ALU:

memory \rightarrow register (Load Operation)

register \rightarrow memory (Store Operation)

"addi" with immediate operand 32bit add
 ↗ register with number immediate 32bit
 32bit integer value 32bit

Register Operands

Immediate Operations

$$f = \$53 + 4$$

(1)

6 addi \$s3, \$s3, 4.

ଏକାଟି register ଏଇ ୨୦୭୨୮ ମେସର ପାଇଁ

integer ইন্টিগার ইন্টেজার

~~oxx~~ addi operation ~~ZT~~

$f = 53 - 4$

6 addi \$S3 , \$S3 , -4

dependency start register overwrite

କବୀ ମାୟ ପା

Q2 $\frac{1}{\sqrt{2}}$ example ?
 center dependency
 $\frac{1}{\sqrt{2}} \pi$ 012-
 register overwrite
 $\frac{1}{\sqrt{2}}$ error

$$\begin{cases} f = g + h \\ a = b + g \end{cases}$$

f → \$52
g → \$52
h → \$53
b → \$54
a → \$55

gives MIPS Code

add \$t₀, \$5.2, \$53. ;

add \$t₁, \$s4, \$s2;

$$f = \pm 53 + 4$$

addi \$s3, \$s3, 4

f = \$ 53 - 4

add1 ← \$53, \$53, -4

Register file \hookrightarrow

Register	Operands
total	32
to	register
216 th	

$\$t_0, \$t_1, \dots, \$t_9 \Rightarrow$ Temporary register \Rightarrow 8 registers 15 to register ($\$t_0 - \t_7)

$\$t_{10}, \$t_{11}, \$t_{12}, \$t_{13} \Rightarrow$ 4 registers 24 to 27 to register for $\$t_8$ and $\$t_9$

$\$s_0, \$s_1, \$s_2, \dots, \$s_7 \Rightarrow$ Saved registers \Rightarrow 16 registers 28 to 35 registers.

Temporary register:

is used for storing temporary values or variables in future to store later value just for write operation.

Saved register:

is used for storing variables or values

in the future to store later value.

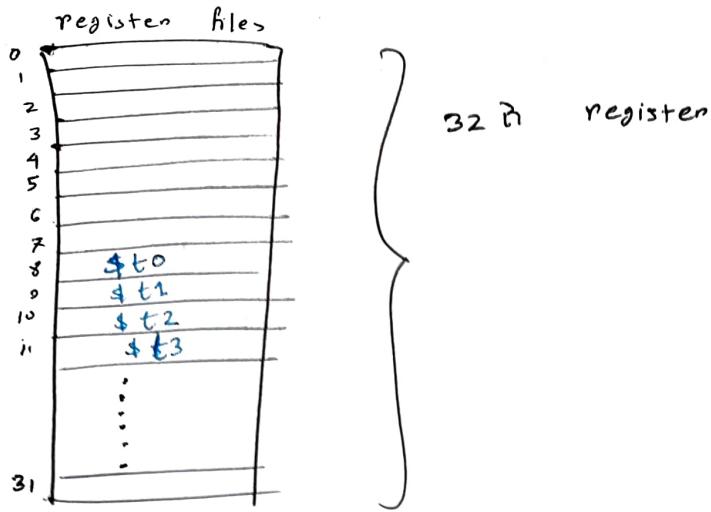
Later value.

Saved register is save value
in over write operation.

$\boxed{\$t_0 = \$8} \Rightarrow$ register file. To 8 to register.

$\boxed{\$t = \$9} \Rightarrow$ register n n n n n

2⁷ 8, 9 960 binary to convert 257.



C-code:

$$a = b + c$$

MIPS: (just concept \Rightarrow $\text{add } \$a, \b, \dots)

add $\underbrace{\$a,}_{\text{destination}} \underbrace{\$b,}_{\text{Source}} \dots + c.$

C-code:

$$f = (g+h) - (i+j)$$

g, h, i, j, f register $\$s_0, \$s_1, \$s_2, \s_3, \dots store
Now write the MIPS code.

→ MIPS code:

add \$t0, \$s0, \$s1

$\left[(g+h) \Rightarrow \text{reg } \$t_0 \text{ consider } g+h \right]$

add \$t1, \$s2, \$s3

$\left[(i+j) \Rightarrow \text{reg } \$t_1 \text{ consider } i+j \right]$

sub \$s4, \$t0, \$t1

ALU operation \Rightarrow ALU \Rightarrow memory \Rightarrow reg \Rightarrow data \Rightarrow
then reg \Rightarrow ALU \Rightarrow result \Rightarrow memory \Rightarrow reg \Rightarrow calculate \Rightarrow then
memory \Rightarrow reg \Rightarrow finally memory \Rightarrow store \Rightarrow

Memory Operand [Example 1] : (Slide - 14) :

C - code:

Code:
 $g = h + A[8]$  *जबकि* memory file से ओर, तो *पूर्व* add res 6272n data  fetch
जबकि *पूर्व* 6272n data  fetch

g in \$s1 ; h in \$s2 ; base address of A in
\$s3 .

Now write the MIPS code for the given c-code,

$$\Rightarrow A[8] \text{ এবং } \text{memory address} = (8 \times 4) + \$53$$

$$= \textcircled{32} + \textcircled{\$53}$$

→ offset → base address

\therefore MIPS Code:

lw \$t0, {32}(\$s3) [\$t0 register > memory
load operator add res over data in-
add \$s1, \$s2, \$t0 fetch करा नियम अनुसृत]

load one stone operator will parameter to,
first parameter to register. 2nd parameter

26cm memory add res.

operator represented as single line

Step 2

547

load from memory into register & write back to store

register from memory to -

data from memory to

read address from

C-code:

$$A[12] = h + A[5]$$

data store to

h in \$s2 ; base address of A in \$s3.

⇒ MIPS / Code:

$$\text{address of } A[5] = (5 \times 4) + \$s3$$

$$\text{address of } A[12] = (12 \times 4) + \$s3 \\ = 48 + \$s3.$$

∴ MIPS code:

lw \$t0, 20(\$s3)

add \$t0, \$t0, \$s2

sw \$t0, 48(\$s3).

Memory සඳහා slot ගැනීමෙන් 8 bit data පිළිබඳ
 යුතු ඇතුළු mips architecture use 32 bit memory

නෑම පිළිබඳ slot ගැනීමෙන් address 32 bit, 1GB,
 මෙයින් memory ගැනීමෙන් size මෙහේ ඇත්?

$$\begin{aligned}
 & \Rightarrow \text{memory size,} = 2^{32} \times 8 \text{ location address combination} \\
 & = 39359738368 \text{ bit} \\
 & = 4294967296 \text{ byte.} \\
 & = 4.29496 \text{ Gb}
 \end{aligned}$$

\$ Zero Register

- ⇒ \$ zero register යන්නේ unique register
- ⇒ \$ zero ගැනීමෙන් value ප්‍රකාශනය constant 0.
- ⇒ Can't be overwritten.
- ⇒ register file ගැනීමෙන් first register 2 \$ zero register.
- ⇒ Used for more operation. example.

Example:

$\$s1 = 7$

$\$t2 = \$s1$

gdb mips code : 268.

add \$t2, \$s1, \$zero

MIPS Register File

MIPS register file contains

32 bits registers
 $= 2^{32} = 4,294,967,296$

⇒ thirty two 32-bits registers

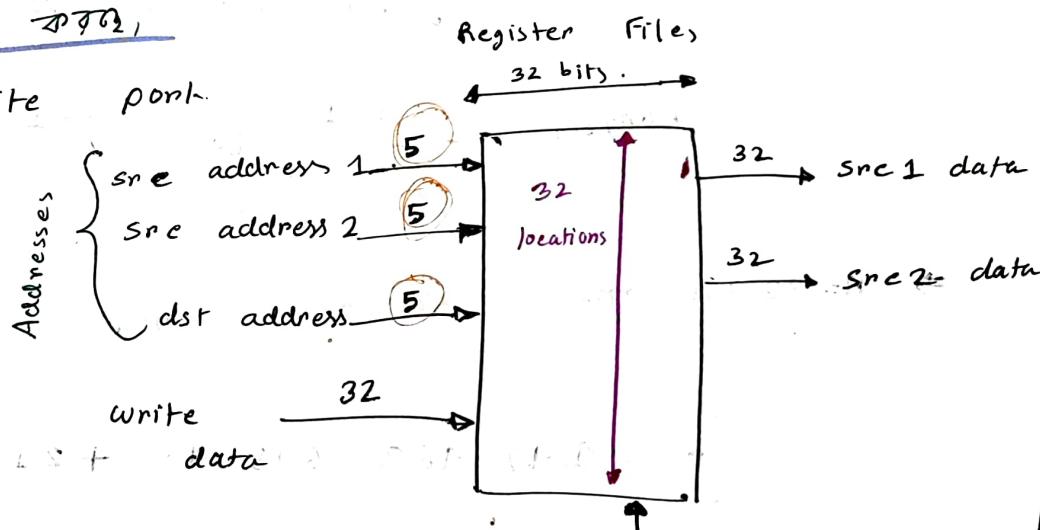
⇒ Two read port

data read info

from 32 register data

collect info

⇒ One write port



add \$s1, \$t0, \$t2,

destination register
dst address

data write info,

then destination info - 1 then

Unsigned binary integer

⇒ any positive binary integer same रूप.

⇒ n-bit system or n-bit number मिहि एवं वाले रखता है।
 range: $[0 \text{ to } +2^n - 1]$ 8 bit एवं 8 bit register
 में कोई नियम नहीं लगता इसका value stored
 जाता है।

मिहि n=8 एवं 01111111 range 0 to $2^8 - 1$ एवं 0 to 255.
 अब 0 एवं 0 to 255 represent करता है 8 bit एवं 8 bit register

Similarly, मिहि n=32 bit एवं 01111111111111111111111111111111 range 0 to 4294967295

2's complement Signed Integer

⇒ n-bit system or number मिहि एवं वाले रखता है। 2's

complement signed integer एवं range:

$$\text{range: } -2^{n-1} \text{ to } 2^{n-1} - 1$$

⇒ n=32 bit एवं उपर्युक्त range:

$$-2^{15} 147 483 648 \text{ to } +2^{15} 147 483 647$$

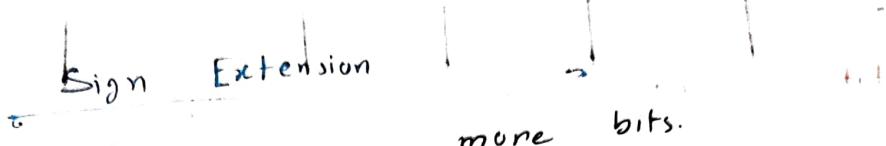
⇒ "Signed" binary एवं first bit एवं MSB मिहि 1

मिहि एवं negative number indicate वाले रखता है।

MSB 0 एवं positive number indicate वाले रखता है।

⇒ Most negative number: 1000000 ... 0000

Most positive number: 011111 ... 1111



⇒ representing a number using more bits.

⇒ example: (8 bit extended to 16 bit).

• +2: 0000 0010 (8bit) → 0000 0000 0000 0010

• -2: 1111 1110 (8bit) → 1111 1111 1111 1110

★ MIPS instructions can be divided into three classes,

⇒ R type

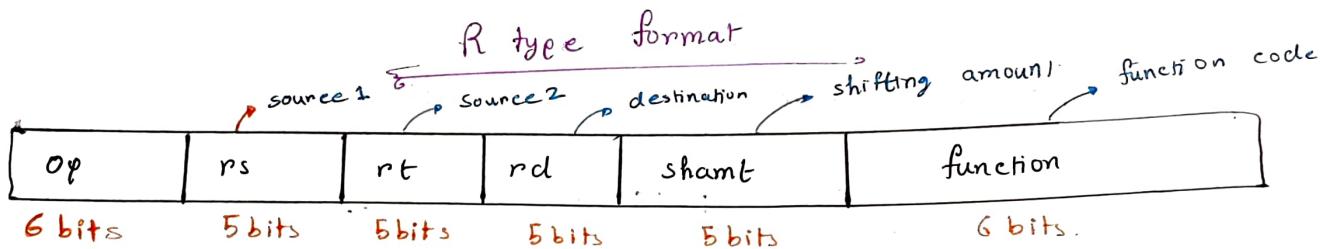
- add, sub, and, or, sll, srl, slt, sltu
- [Arithmetic operation].

⇒ I type:

- lw, sw, addi, beq, bne
- branch not equal.
- slti, sltui
- branch equal
- set less than i

⇒ J type:

- j (Jump)



R type କିମ୍ବା ରେଜି ଓଡ଼ି ଓଡ଼ିକ୍ସ୍‌କୋଡ୍ " 000 000 " ହେଲା ।

ଏହା କିମ୍ବା shifting କିମ୍ବା କିମ୍ବା shamt = 0 ହେଲା

(ଯତେ bit ହେଲା)

ଏହାରେ କିମ୍ବା bit shifting କିମ୍ବା କିମ୍ବା corresponding binary କିମ୍ବା

binary (ଯତେ bit ହେଲା)

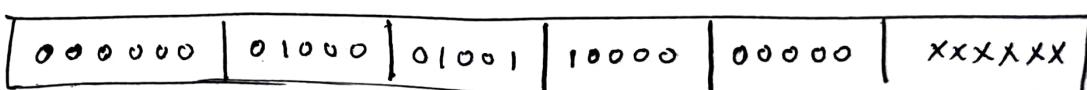
function କିମ୍ବା କିମ୍ବା corresponding value କିମ୍ବା କିମ୍ବା function =
 ଏହାରେ corresponding binary କିମ୍ବା କିମ୍ବା କିମ୍ବା 6 bit
 don't care - (XXXXXX)

R type ଏହା register ଏହା register ଏହା operation ହେଲା

add \$t0, \$t1.



register କିମ୍ବା corresponding index.



I-Type format



⇒ मध्ये Opcode असा 2bit ठारले तरी. लिखत वरपास
 "xxx xxx" लिखू.

⇒ base address वरपास rs लिखत, offset part तुक्क
 constant / address part # constant आणि range
 -2¹⁵ to +2¹⁵ - 1

add \$s0, \$s1, 2

X	\$s1 (17)	\$s0 (16)	2
---	--------------	--------------	---

xxxxxx	10001	10000	0000 0000 0000 0010
--------	-------	-------	------------------------

lw 4 to, 12 C(\$s0)
 sw \$t1, 24 C(\$s1)
 pt.

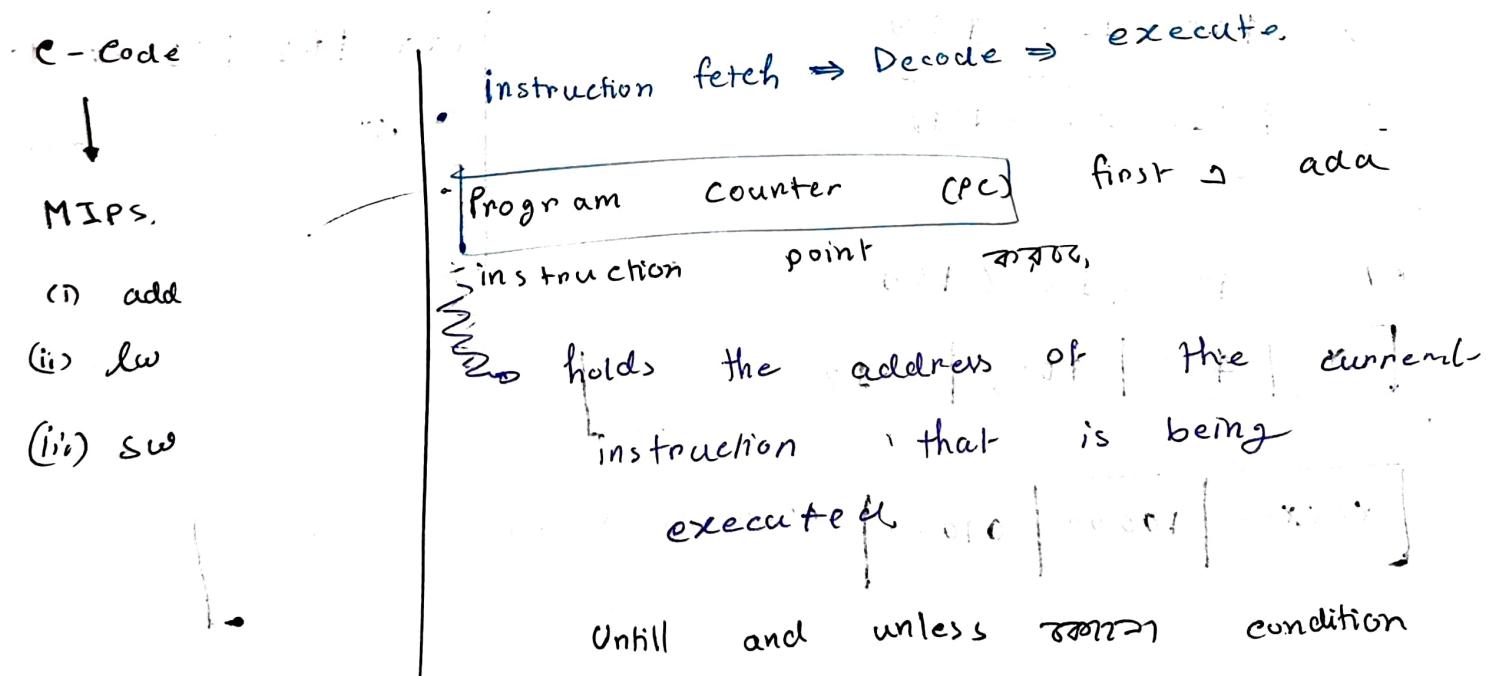
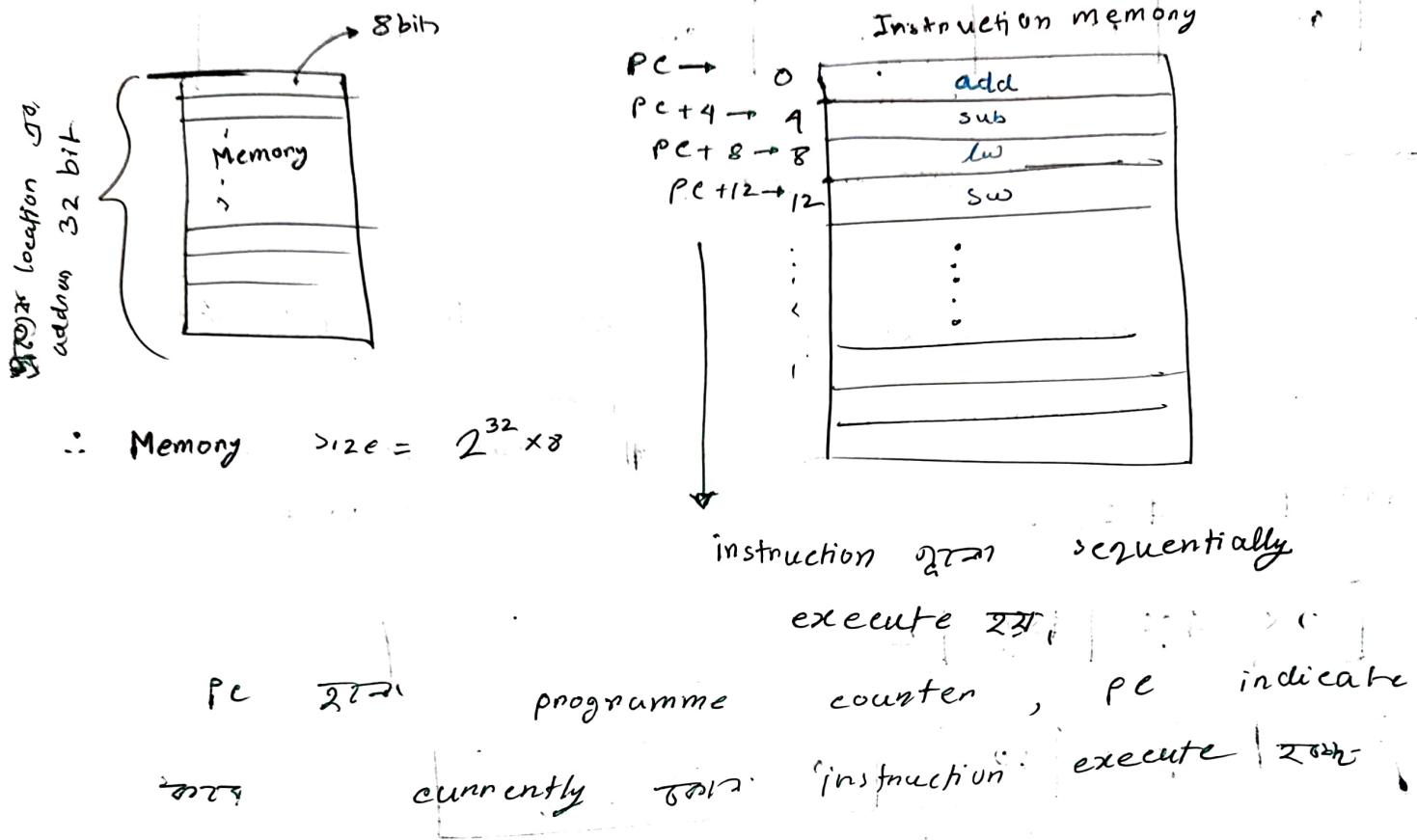
add \$t1, \$s1, 20 C(\$s3)
 ↳ अंग 3 I-type format

• lw \$to, 12 (\$s0)

X	\$s0 (16)	\$to (8)	12
---	--------------	-------------	----

xxxxxx	10000	01000	0000 0000 0000 1100
--------	-------	-------	------------------------

Conditional Operations and Branch Address.



(PES)

$PC + 4$ \rightarrow increase PC ; as memory is at slot.

for 32 bit PC .

PC (Program Counter) \rightarrow register.

PC first \rightarrow initial instruction \rightarrow address point \rightarrow PC \rightarrow first instruction \rightarrow point \rightarrow PC \rightarrow next instruction \rightarrow Then \rightarrow PC \rightarrow current value \rightarrow ALU \rightarrow Then \rightarrow add \rightarrow next current value \rightarrow \rightarrow \rightarrow current instruction \rightarrow address point \rightarrow PC \rightarrow

I-format:

$beq \$a, \$b, L1$.

$\hookrightarrow \$a \rightarrow \$b \rightarrow$

check \rightarrow , equal \rightarrow

\hookrightarrow jump \rightarrow

value

equal for
L1 function

$bne \$8, \$9, L2$

$\hookrightarrow \$8 \rightarrow \$9 \rightarrow$ value not equal \rightarrow

\rightarrow L2 function \rightarrow jump

\rightarrow

Calculator uses neg from 2's complement ~~uses~~ ~~not~~

~~not~~ not first 1's complement.

+2 \Rightarrow 0000 0010 ;
8 bit

1's complement of (+2) \Rightarrow 1111 1101

2's complement of (-2) \Rightarrow 1111 1110

$\therefore -2 \Rightarrow$ 1111 1110
8 bit representation

+2 (8 bit extended to 16 bit)

0000 0000 0000 0010

-2 (8 bit extended to 16 bit)

1111 1111 1111 1110

1110

addi \$50, \$t1, -4 → do 2's complement first, and then use sign extension to get the 16 bit representation of "-4"

এবং এই -4 এর 16 bit representation ই I type format এর constant এর address field এ রাখা হবে,

branch instruction

→ Conditional branch
 • যদি কোথা condition এর উপর base
 কর্তৃ Jump করে কোথা location এ যাই,

→ Unconditional branch
 • যদি কোথা condition ইন্দুষ্ট্রি ; (Jump)
 কর্তৃত বলতে,

C-code:

```

if (a == b) {
    a = b + 1
}
else {
    a = b + 2
}

```

so ~~if~~ MIPS check ~~for~~ ~~বলো~~ equal ~~বলো~~ check ~~বলো~~ ~~বলো~~
 then ~~if~~ MIPS check ~~বলো~~ not equal ~~বলো~~ check ~~বলো~~

$a = b + 1$

}

else {

$a = b + 2$

}

a is stored in \$S1

b is stored in \$S2

assume এভিয়ে else function রে computer

random রেখি number 2 generate করো।

$$\therefore (2 \times 4) = 8 \text{ slot}$$

মনে করো 1st instruction রেখি 2nd instruction
 8 টি slot skip করো এবং else function -

jump ~~করো~~ ~~এভিয়ে~~

C - code :

```

if (a != b) {
    a = b + 1
}
else {
    a = b + 2
}

```

a is stored in \$S1

b is stored in \$S2

MIPS code :

1. beq \$S1, \$S2, Else
2. addi \$S1, \$S2, 1
3. j Exit
12. Else: addi \$S1, \$S2, 2
16. Exit.

Stored program Computers:

- Instructions and data both are represented in binary.
- Instructions and data both are stored in memory.
- Instructions and data both are stored in programs.
- Programs can operate on programs like : compiler, linkers
- Binary compatibility allows compiled programs to work on different computers.
- Standardized ISA.

Operation	MIPS
Shift left	sll
shift right	srl
Bitwise AND	and, andi
Bitwise OR	or, ori
Bitwise NOT	nor
Set if less than	slt, slti

register এর ২০৮২৫ কাটা integer value add করে ২৪৯
 "addi "

e-code:

```

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

```

f, g, h, i, j are stored correspondingly in

~~\$50, \$51, \$52, \$53, \$54~~

MIPS:

PC ← 0bne \$S3, \$S4, (\$Else)
PC ← add \$S0, \$S1, \$S2

PC+8 → ~~s~~ j exit

PC-112 Else:

PC+16 → 16 sub \$50, \$51, \$52.

PC + 20 → 20 Exit.

Computer else block এর অন্ত রেখা- random integer
 generate করে (৫৮৫২) Suppose, 2 generate করে
 দিচ্ছা, ০১৩ ১১৬ তার এর currently টেক্স ইনস্ট্রুক্ষন
 ২ অংশ ০১৩ ১১৬ ২ লাইন $(2 \times 9 = 8$ slot) skip

23 branching condition Br^{102} ,

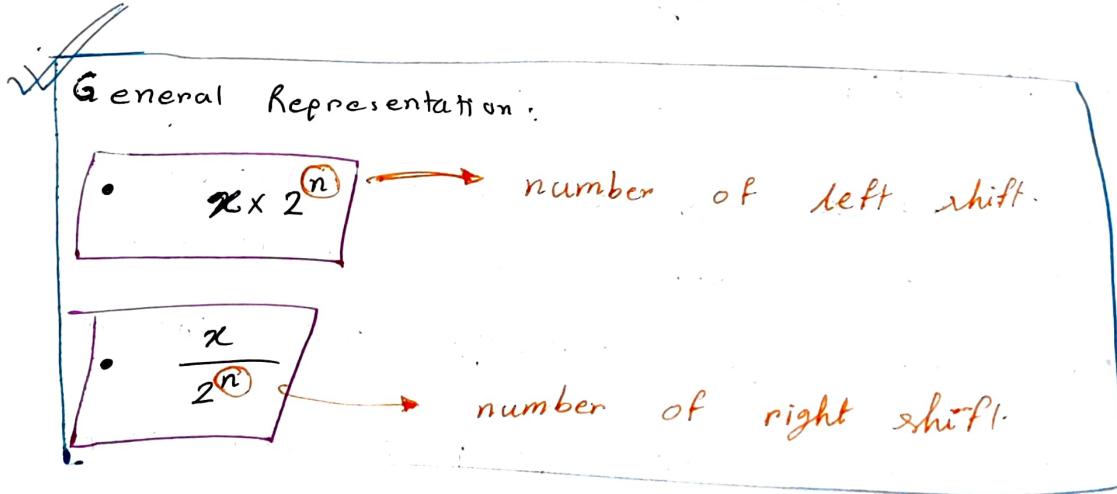
`beg` } i-type instruction
`bne.` }

5 bit register

$4 \rightarrow 00100$ 1 bit left shift 01000 1 bit left shift 10000

4 8 16

line skip
or indicate
means,
 $4 \times 2 = 4 \times 2^1 = 8$ number of shift
 $4 \times 4 = 4 \times 2^2 = 16$ 2^1 power 2^2



32 bits architecture to $x \times (2^1)$ number

multiplication by 2^2 in 4 2^3 , 2^4

search data memory in 4 slot

means 256 slots, 16 bits 2^{16} , $2^{16} \times 2^2 = 2^2$

multiply 256 2^{16} 32 bit architecture

256 2^{16} 2^{16} ~~256~~ 2 bit left shift

256,

256 multiply 256 1 bit left shift 256,

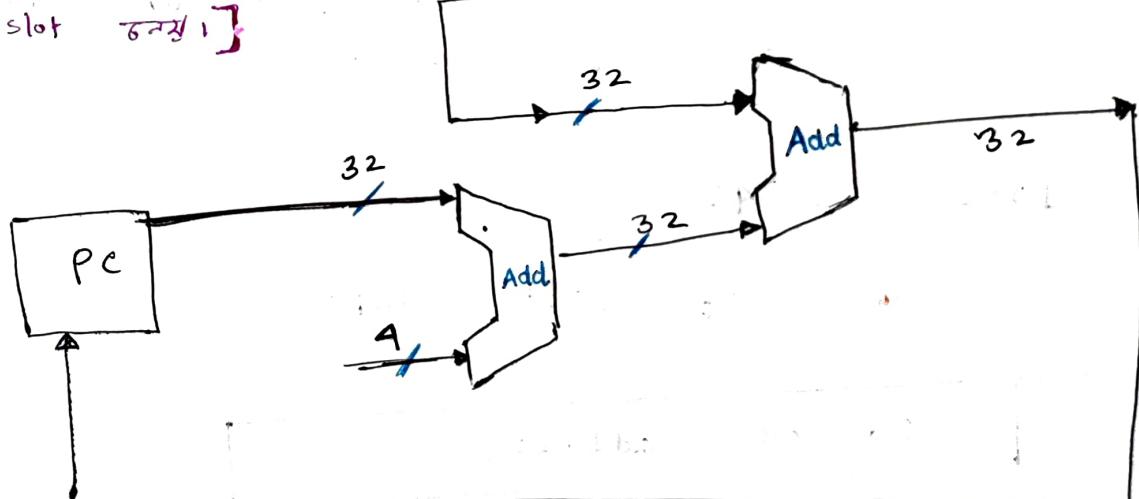
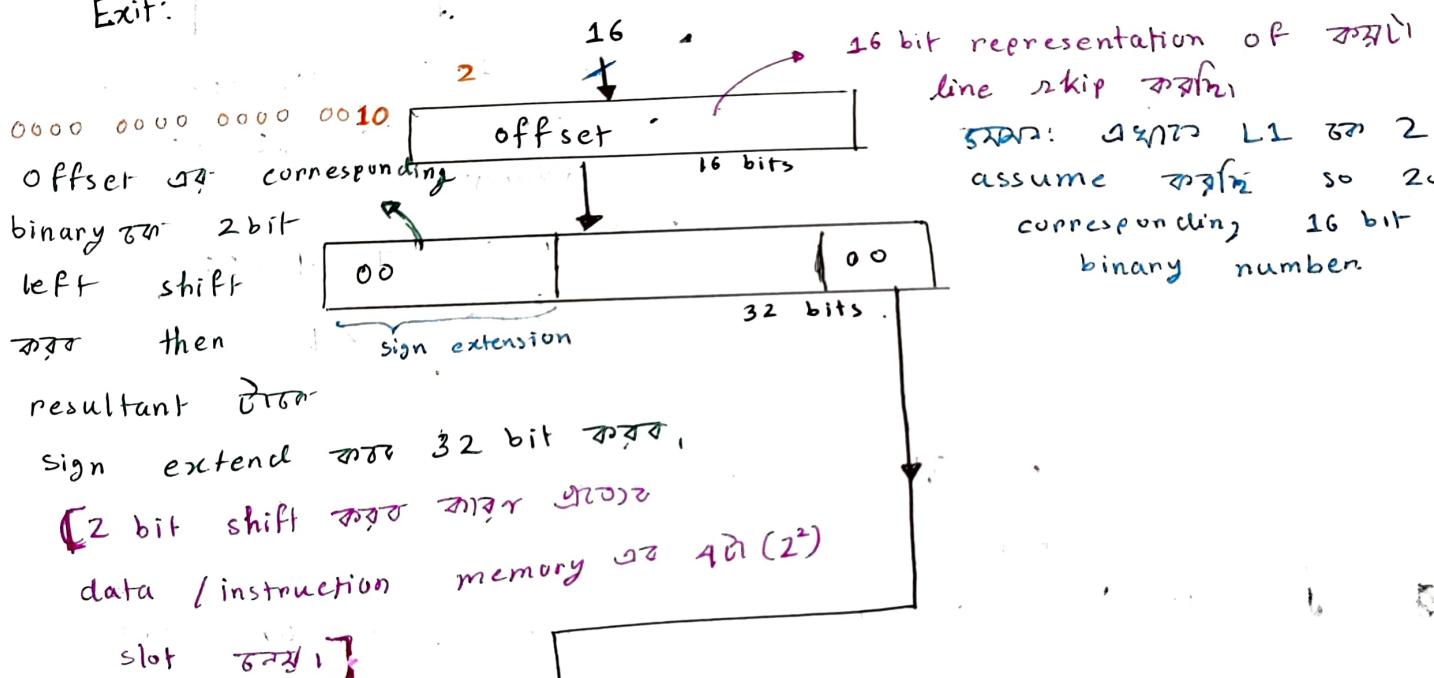
beg \$S1, \$S2
 add
 j Exit
 L1:
 add
 Exit:

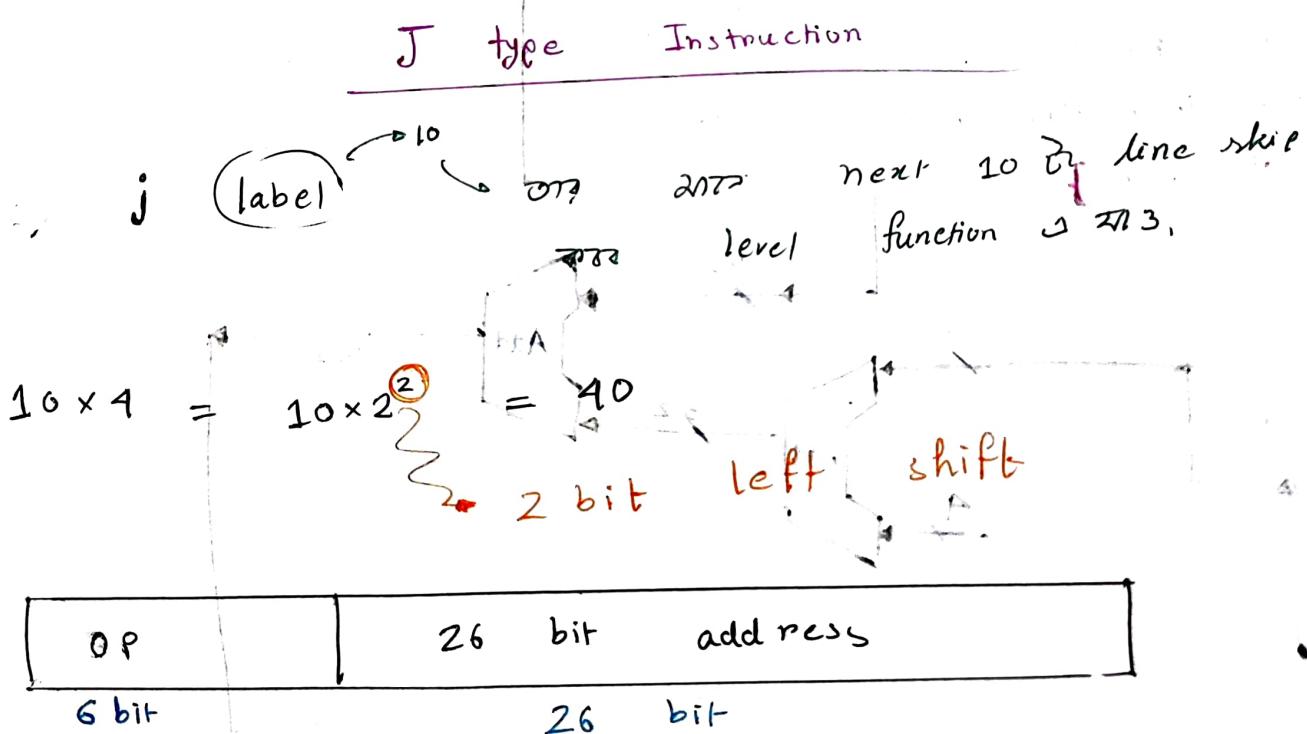
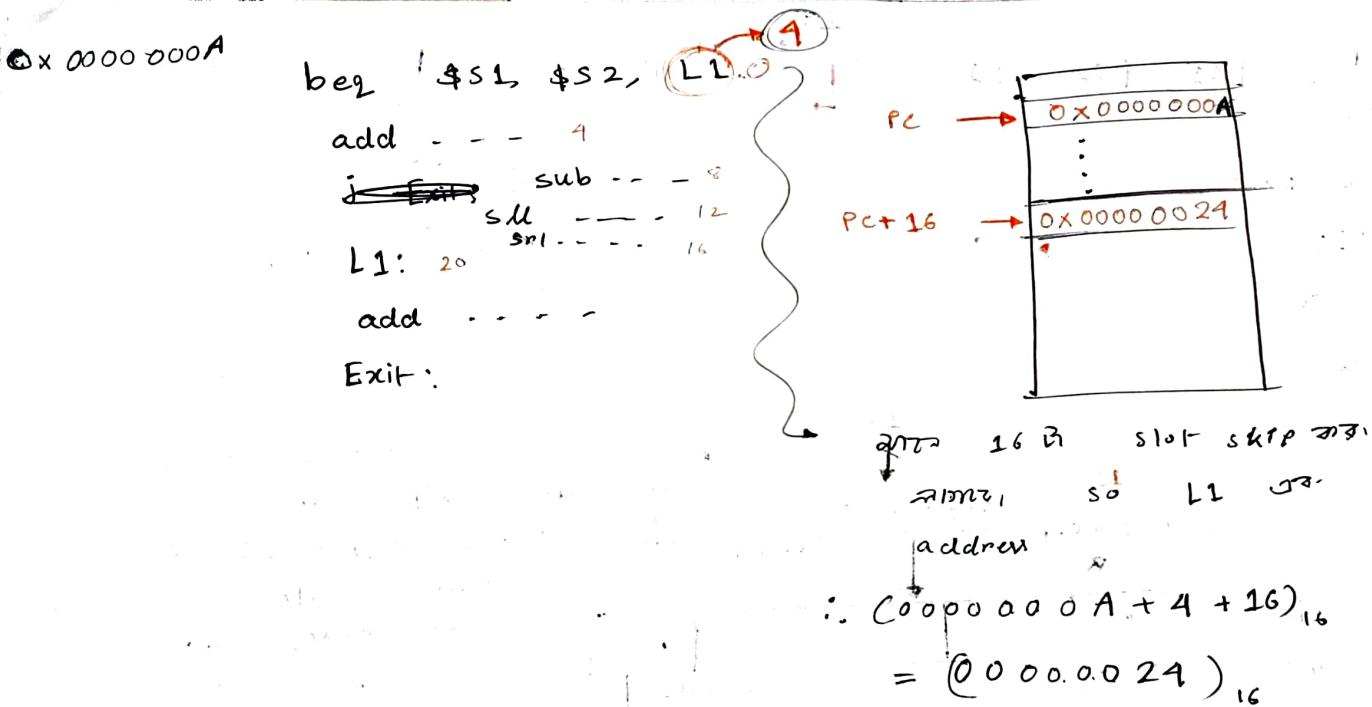
branch address calculate করুন

2nd inst 2 slot এর L1

একটি

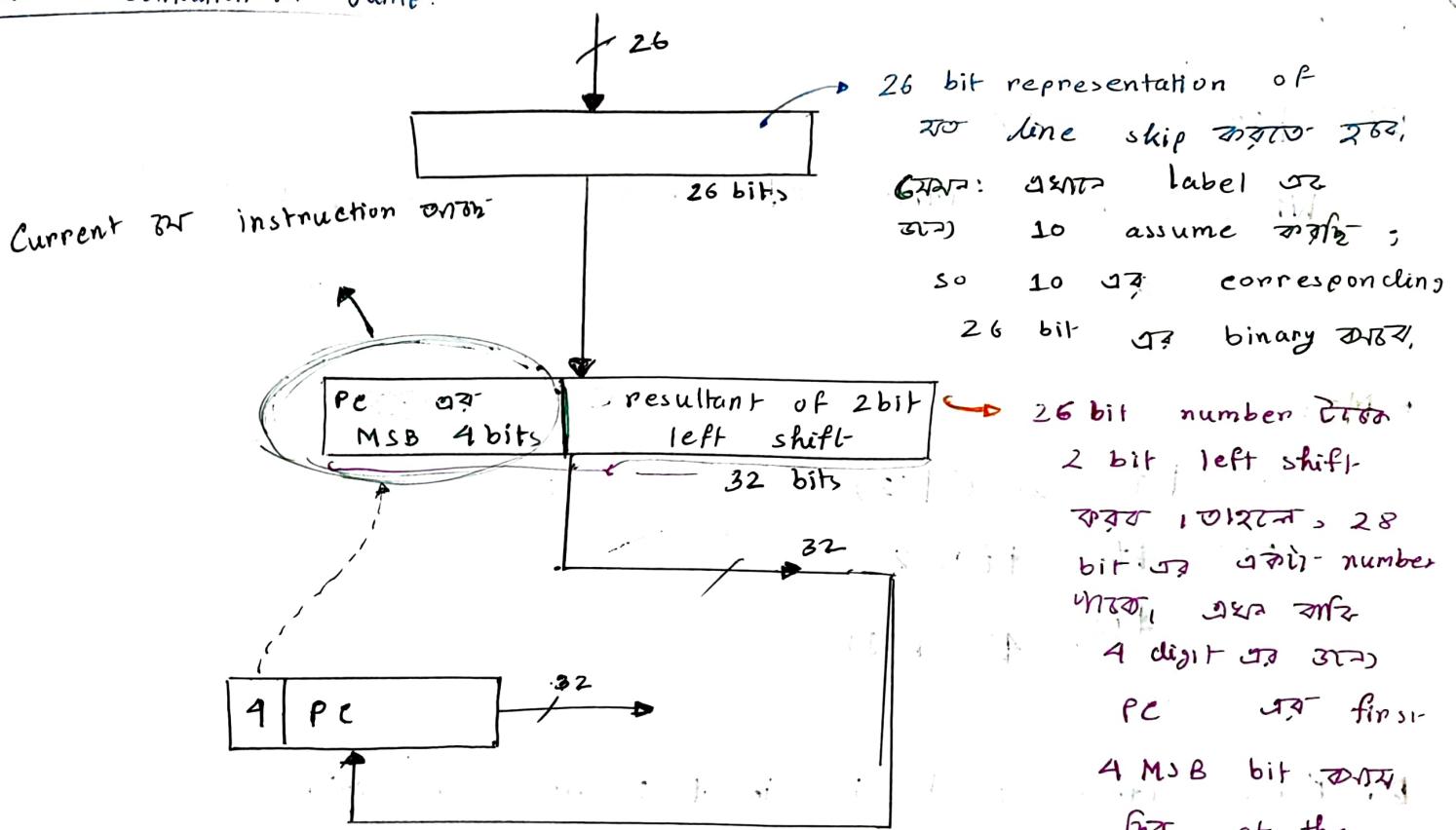
Current inst 8 slot এর L1





Opcode 2444 244444
 don't care "XXXXXX" 244444

Branch Destination for Jump:



Given: Address label or
assume ;
so 10 corresponding
26 bit as binary ;

26 bit number ;
2 bit left shift
करने का रास्ता, 28
bit का एक 4-digit
में, एक अंक
4 digit का उस
PC का first
4 MSB bit की
होती है वह
beginning of that
28 bit number.

in J type instruction we didn't need ALU.
any ALU. Because, J type instruction do not
use any kind of operation for
use ALU for calculating address
Σ known as concatenation method

MIPS Code Practices

■ $B[10] = A[6] + 2$

Where base address of B and A are \$51 and \$2 respectively

MIPS:

lw \$t0, 20(\$52)

add \$t0, \$t0, 2

sw \$t0, 40(\$51)

■ Base address of A is in \$50. Write the MIPS code for the given set of c-code,

if ($A[3] \neq A[6]$) {

if ($A[3] == 0$) {

$A[3] = A[3] + 2;$ }

else {

$A[6] = A[6] / 16;$ }

}

else {

$A[6] = A[6] * 8$ }

$\frac{x}{2^4}$ and sm

2^3 ~ sll

⇒ MIPS code:

lw \$t0, 12(\$s0) # A[3]

lw \$t1, 24(\$s0) # A[6]

bge \$t0, \$t1, L1

bne \$t0, \$t1, L2

addi \$t2, \$t0, 2 # A[3] + 2

sw \$t2, 12(\$s0)

j Exit

L2:

~~addi \$t3, \$t1~~ # 4 bit right shift

srl \$t3, \$t1, 4

sw \$t3, 24(\$s0)

j Exit

L1: # 3 bit left shift

sll \$t4, \$t1, 3

sw \$t4, 24(\$s0)

Exit:

x, y, z are stored in $\$s_0, \$s_1, \$s_2$.

$$x = \underbrace{2y}_{\$t_0} + \underbrace{65z}_{\$t_1} + 10$$

→ MIPS:

add $\$t_0, \$s_1, \$s_1$ # $2y$

sll $\$t_1, \$s_2, 6$ # 6 bit left shift

add $\$t_1, \$t_1, \$s_2$ # $65z$

add $\$t_0, \$t_0, \$t_1$ # $2y + 65z$

addi $\$s_0, \$t_0, -10$

$$64z + z = 65z$$

$$z \times 2^6 = 64z$$

6 bit left shift.

More Conditional Operation:

- SLT (R type)

- SLTI (I type)

SLT → Set less than

set less than 1

Greater than check

SLT to SLTR

use $\overline{Z_{01}}$,

$\$s_3 \geq \s_4

greater than $\overline{Z_{01}}$ $\overline{Z_{10}}$ $\overline{Z_{11}}$

register $\hookrightarrow 0$ store $\overline{Z_{01}}$

$\$s_3 < \s_4

less than $\overline{Z_{01}}$ $\overline{Z_{10}}$ $\overline{Z_{11}}$

register $\hookrightarrow 1$ store $\overline{Z_{01}}$

Format (contd).

SLT (rd) rs, rt
destination

↳ if $(rs < rt)$ then $rd = 1$ else $rd = 0$

SLT (rt) , rs, constant
destination

↳ if $(rs < \text{constant})$ then $rt = 1$ else $rt = 0$

• Use in combination with "beg" and "bne"

SLT \$t1, \$s3, \$s4

সম্ভব $\$s3 \geq \$s4$ হল।

অর্থাৎ $\$t1 = 0$ সেব।

হল।

অর্থাৎ, $\$s3 < \$s4$ হল।

$t1 = 1$ save হচ্ছে।

set $t1 = 1$ assign $\$t1$

check $\$t1$ \rightarrow SLT $\$t1$ $\$s4$

reset $t1 = 0$ assign $\$t1$

consider $\$t1$.

$\$t1$ value \rightarrow $\$t2$

করুন

an. int

$\$t2$

$\$t2$

C - code:

```

if (a < b) {
    a = a+1 ;
}
else {
    a = a+2 ;
}

```

Main code is L; so compare
 if zero then greater than
 equal or zero compare
 else,

MIPS:

```

slt $t1, $s0, $s1
beq $t1, $zero, Else:
addi $s0, $s0, 1
j Exit

```

Else:

```
addi $s0, $s0, 2
```

Exit:

$$\$t0/16 = \$t0/2^4 \rightarrow 4 \text{ bit right shift}$$

signed

comparison \Rightarrow slt, sltr

unsigned

comparison \Rightarrow sltu, sltur

■ Why not blt, bge, etc?

- ⇒
- Hardware for $<$, \geq , ... slower and complex than $=$, \neq .
 - Combining with branch involves more work per instruction, requiring a slower clock.
 - All instructions are penalized.

■ Write the MIPS code for the following code,

$x = A[i] + 2$
where, x , ~~A~~, base address of A and i are
in $\$s_1$, $\$s_2$ and $\$s_3$ respectively.

⇒

~~add \$t0, \$s3, 2~~ # multiplying (~~*4~~) so 2^{31} -
~~left shift.~~

add \$t0, \$s2, \$t0 # finding the mem. address
of ' $A[i]$ '

lw \$t1, 0(\$t0) # retrieving the value of $A[i]$ to
 $\$t1$.

addi \$s1, \$t1, 2.

$$A[B[i]] = x$$

Where base address of A and B are in \$s1 and \$s2 respectively and x and i are in \$s3 and \$s4.

⇒

```

sll $t0, $s4, 2
add $t0, $t0, $s2
lw   $t1, 0($t0)
sll $t0, $t1, 2
add $t0, $t0, $s1
sw   $s3, 0($t0)

```

Formula:

- Memory Address for data in array = base address + (index $\times 4$)
- Branch address = PC + 4 + (offset $\times 4$)
- Jump address = PC (MSB 4 bits) + (offset $\times 4$)
 - offset 23
26 bits rep then
 - 2 bit left shift
 - then PC is MSB
 - 1 bit at the beginning.

R type:

sll (\$t0) (\$s1) 2
 rd rt shamt

000000	0	\$s1	\$t0	2	xxxxxx
OP	rs	rt	rd	shamt	funct

000000	00000	\$17	\$8	00010	xxxxxx
--------	-------	------	-----	-------	--------

(P-T-U)

a and b are stored in \$s0 and \$s1

C - code:

```
if (a > b) {  
    a = a + 1 ;  
}  
else {  
    a = a + 2 ;  
}
```

Main code $a >$; so
compare बड़ा है तो
less compare नहीं,

⇒ MIPS:

slt \$t1, \$s0, \$s1

bne \$t1, \$zero, Else

$\$s0 < \$s1$ तो
 $\$t1 = 1$

addi \$s0, \$s0, 1

j Exit

Else:

addi \$s0, \$s0, 2

Exit:

Op	rs	rt	rd	shamt	funct
000000	000000	10001	01000	00010	XX XXXX

sll 003 5pl operations

000000

01000

rs

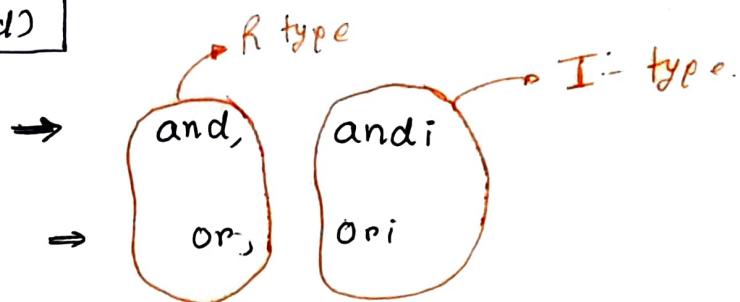
20000000

0 264,

c-code:

$$g = h + A [8]$$

Bitwise AND



Bitwise NOT

nor

r format
register, int, andi
int or andi

and \$t0, \$t1, \$t2
 rd rs rc

$$\begin{aligned} \$t1 &= 1110 \\ \$t2 &= 0101 \\ \therefore \$t0 &= 0100 \end{aligned}$$

andi \$t0, \$t1, 2

$$\begin{aligned} \$t1 &= 1011 \\ \$t2 &= 0010 \\ \therefore \$t0 &= 0010 \end{aligned}$$

Or \$t0, \$t1, \$t2

$$\begin{aligned} \$t1 &= 1110 \\ \$t2 &= 0100 \\ \therefore \$t0 &= 1110 \end{aligned}$$

ori \$t0, \$t1, 2

$$\begin{aligned} \$t1 &= 1011 \\ \$t2 &= 0010 \\ \$t0 &= 1011 \end{aligned}$$

a nor b = Not (a or b)

out a or b
 or operation
 then 3's complement

Not zero (3's)

nor \$t1, \$t2, \$zero.

$\$t2 = 1011$

$\$zero = 0000$

$\therefore \$$ or operation $\text{or } m = 1011$

Ans will not zero.

$\therefore \$t1 = 0100$

nor $\$t1, \$t2, \$t3$

$\$t2 = 0100$

$\$t3 = 1000$

or operation $\text{or } m = 1100$

Not $\text{or } m$, $\$t1 = 0011$

Loops

while loop:



C - code:

```
while ( save [i] == k ) {
```

```
    a = a + 2;
```

```
    i += 1;
```

```
}
```

i, k, a are stored in $\$s3$, $\$s5$ and $\$s4$

respectively and base address of save

is $\$s6$.

MIPS Code:

Loop:

sll \$t0, \$s3, 2

add \$t0, \$t0, \$s6 # Memory address of save [i]

lw \$t1, 0(\$t0)

bne \$t1, \$s5, Exit

addi \$s4, \$s4, 2

addi \$s3, \$s3, 1

j loop # loop iterate ~~again~~ ~~else~~

Exit;

For loop:

for (int i=0 ; save [i] > k ; i++) {

a = a+2

}

i, k, a are stored in \$s3, \$s5 and \$s4.

respectively. Base address of save is in \$s6.

MIPS Code:

add \$s3, \$zero, \$zero # i=0 assign করা হচ্ছে initially

loop:

· sll \$t0, \$s3, 2

add \$t0, \$t0, \$s6

lw \$t1, 0(\$t0)

slt \$t2, \$t1, \$s5

bne \$t2, \$zero, Exit

addi \$s4, \$s4, 2

addi \$s3, \$s3, 1

j loop

প্রারম্ভ, main code o
> then < এর পর
compare করা, less than
হচ্ছে \$t2 গুরুত্বের 1 store
হচ্ছে

Exit: