Assignment on Data path & memory

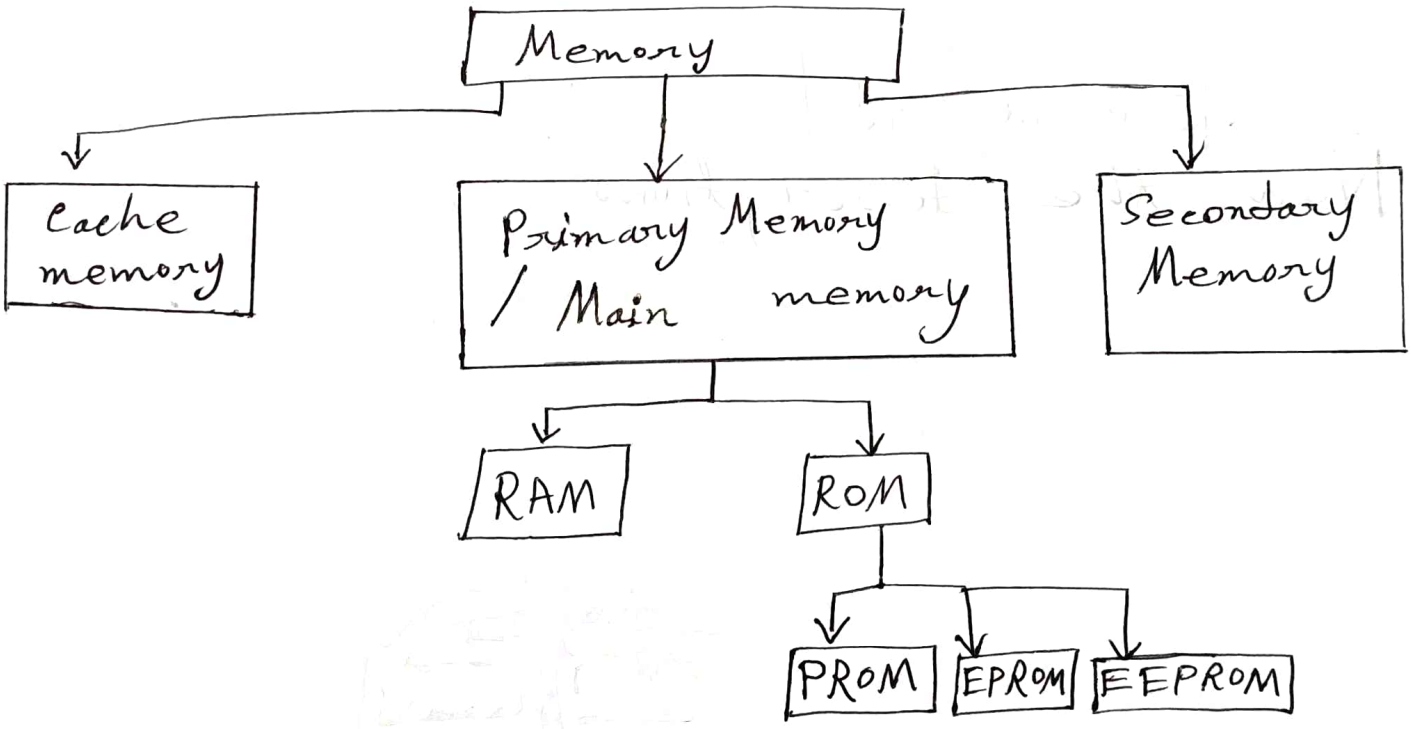
Course: CSE 317

Name : Nianul Hasan
ID : 1720102 6
Sec : A

Q.1 Ans:

: memory or computer memory is the storage space in the computer, where data is to be processed and instructions required for processing are stored.

Now types of memory:



Here, RAM & cache memory, are Volatile memory. ROM & secondary memory, are Non-Volatile memory.

Descriptions;

Cache Memory:

Cache memory is a small, fast memory that acts as a buffer for a slower, larger memory.

Cache memory is used to hold those parts of data and program which are most frequently used by the CPU.

Primary Memory: (Main memory)

Primary memory also called main memory, used to hold programs while they are running. It is divided into two subcategories, RAM and ROM.

RAM :

Random Access Memory (RAM) is also called as read write memory or the main memory.

It is a volatile memory as the data loses when the power is turned off.

RAM is further classified into two types :

1. SRAM
2. DRAM

1. SRAM : (Static Random Access Memory)

Value is stored on a pair of inverting gates that will exist indefinitely as long as there is power, which is why it is called static.

It is very fast but takes up more space than DRAM - 4 to 6 transistors per bit.
SRAM used for cache.

2. DRAM : (Dynamic Random Access Memory)

Value is stored as a charge on capacitor that must be periodically refreshed, which is why it is called dynamic.

It is very small, 1 transitor per bit. DRAM, slower than SRAM. DRAM is used for main memory.

ROM :

Read only memory (ROM), stores crucial information essential to operate the system, like the program essential to boot the computer.

ROM is futher classified into 3 types:

1. PROM, 2. EPROM, 3. EEPROM.

1. PROM: (Programmable read-only memory)

It can be programmed by user. Once programed, the data and instructions in it cannot be changed.

2. EPROM: (Erasable Programmable read only memory)

It can be reprogrammed.

3. EEPROM: (Electrically erasable programmable read only memory)

The data can be erased by applying electric field.

Secondary memory:

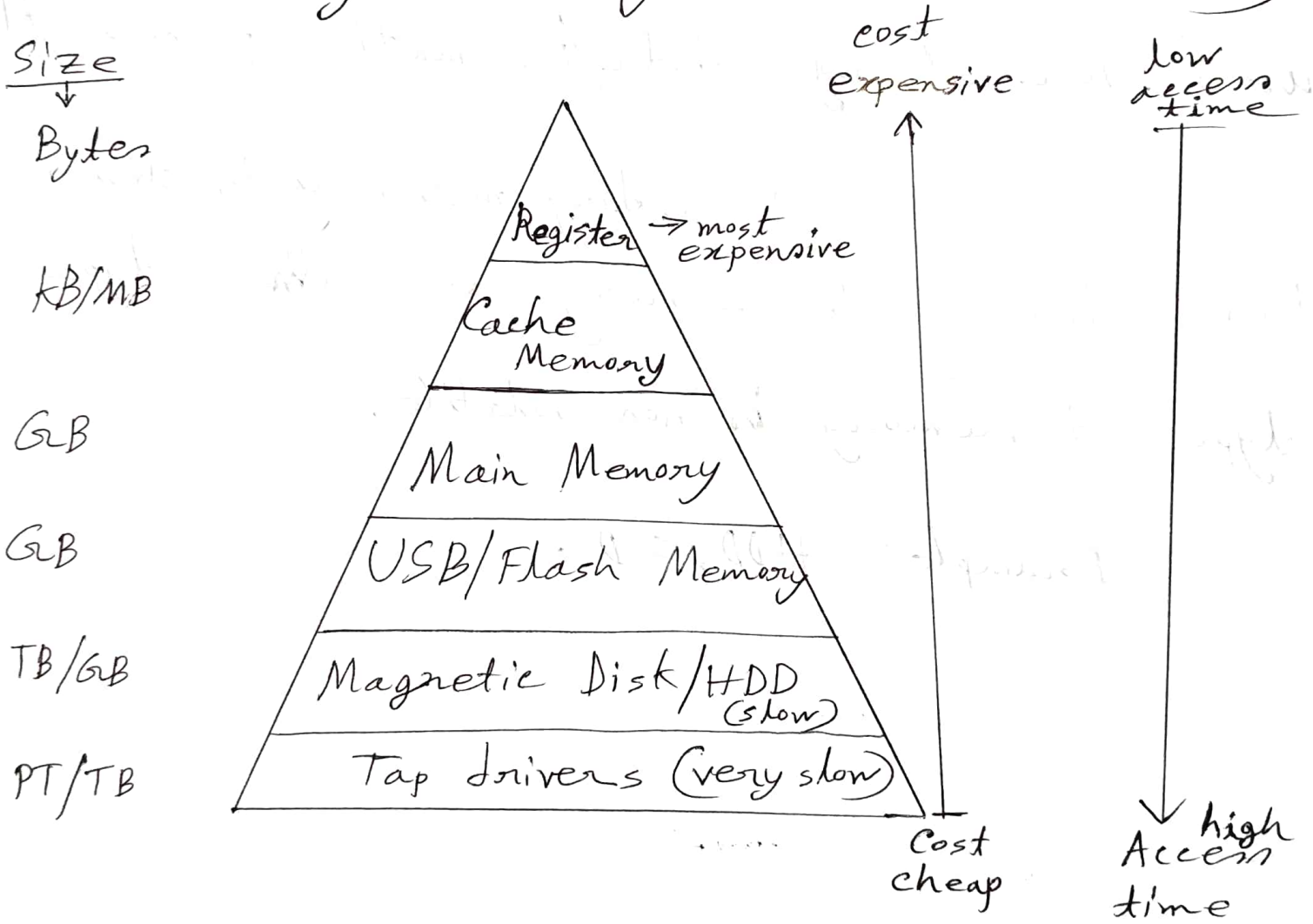
The secondary memory is used for storing data/information permanently.

Secondary memory is slower than the main memory and this type type of memory is non-volatile.

Example: HDD, SSD.

Q.2 Ans:

Memory Hierarchy: (levels of memory)



8.3 Ans:

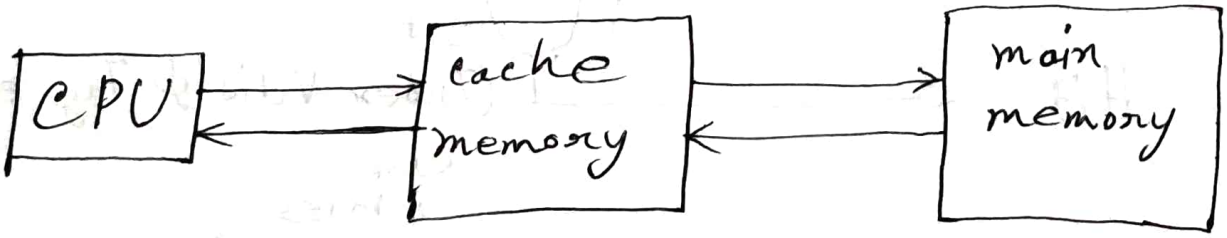
Steps of Cache memory working procedure:

1. CPU initially looks in the cache for the data it needs.

2. If the data is there, It will retrives from here and process it.

3. If the data is not there, then cpu access the system memory and then put a copy of the new data in the cache memory before processing it.

4. Next time if the cpu needs to access the same data again, It will just retrieve the data from the cache memory instead going through the whole loading process



8.4 Ans:

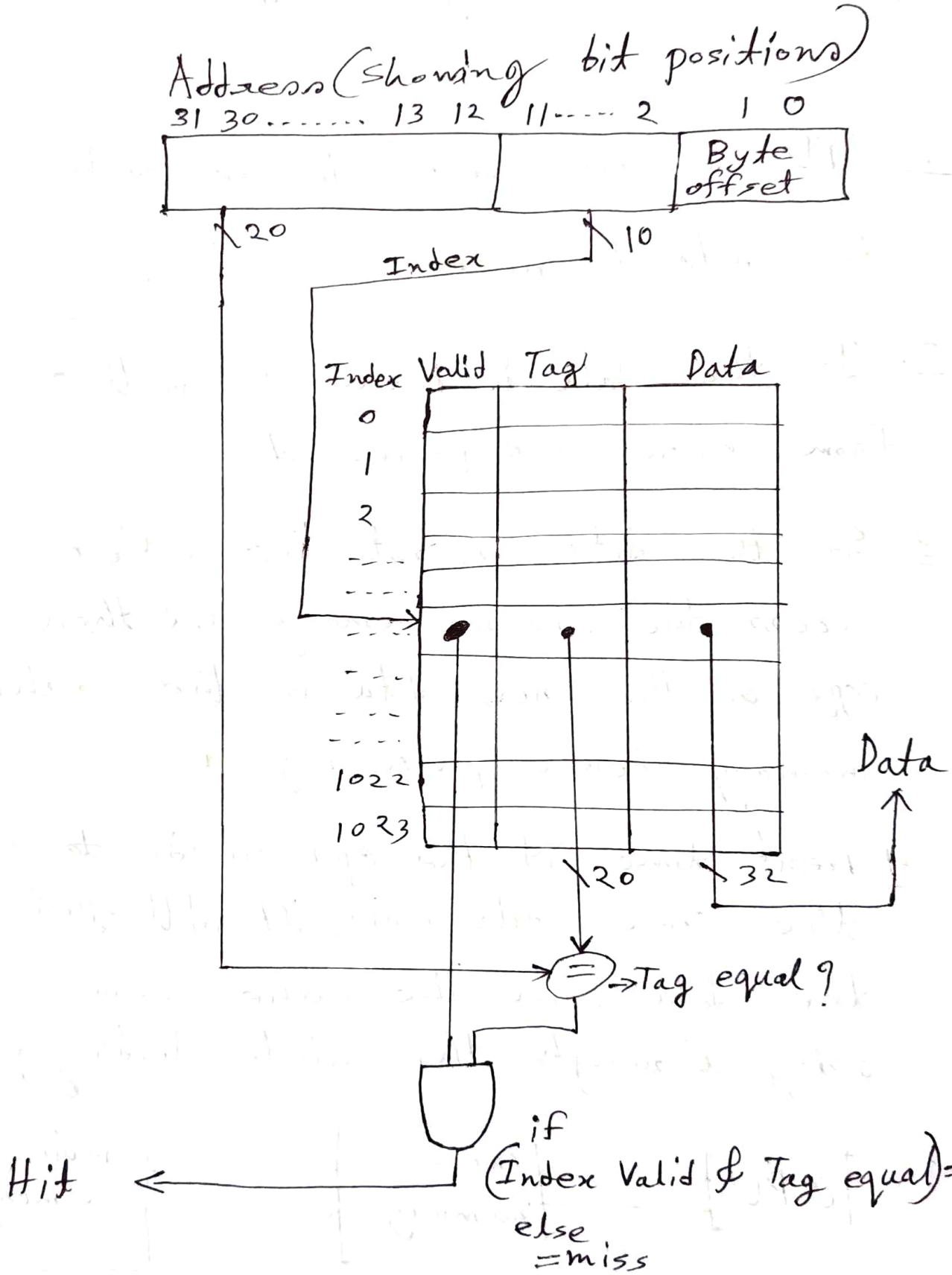


figure: Hit/miss hardware implementation.

Q. 5 Ans:

Cache:

Cache or cache memory is a small fast memory that acts as a buffer for a slower, larger memory.

Memory hierarchy:

Memory hierarchy is a structure consists of multiple levels of memory with different speeds and sizes.

Direct-mapped cache:

Direct-mapped cache structure is a cache structure in which each memory location is mapped to exactly one location in the cache.

Tag:

A field in cache memory, used for a memory hierarchy that contains the address information required to identify whether the associated block in the hierarchy corresponds to a requested word.

Hit time:

The time required to access a level of the memory hierarchy, including the time needed to determine whether the access is a hit or a miss.

Miss rate:

The fraction of memory accesses not found in a level of the memory hierarchy.

Block:

The minimum unit of information that can be either present or not present in the two-level

hierarchy.

Hit :

If the data requested by the processor appears in some block in the upper level, this is called a hit.

Miss :

If the data requested by the processor does not appear in some block in the upper level, this is called a hit.

8.6 Ans:

Here, 32 words cache,
512 words main memory,
block size 4 words

∴ number of bits for physical address, 9 bits

$2^9 = 512$

∴ main memory blocks $= \frac{512}{4} = 128$

∴ cache lines $= \frac{32}{4} = 8$

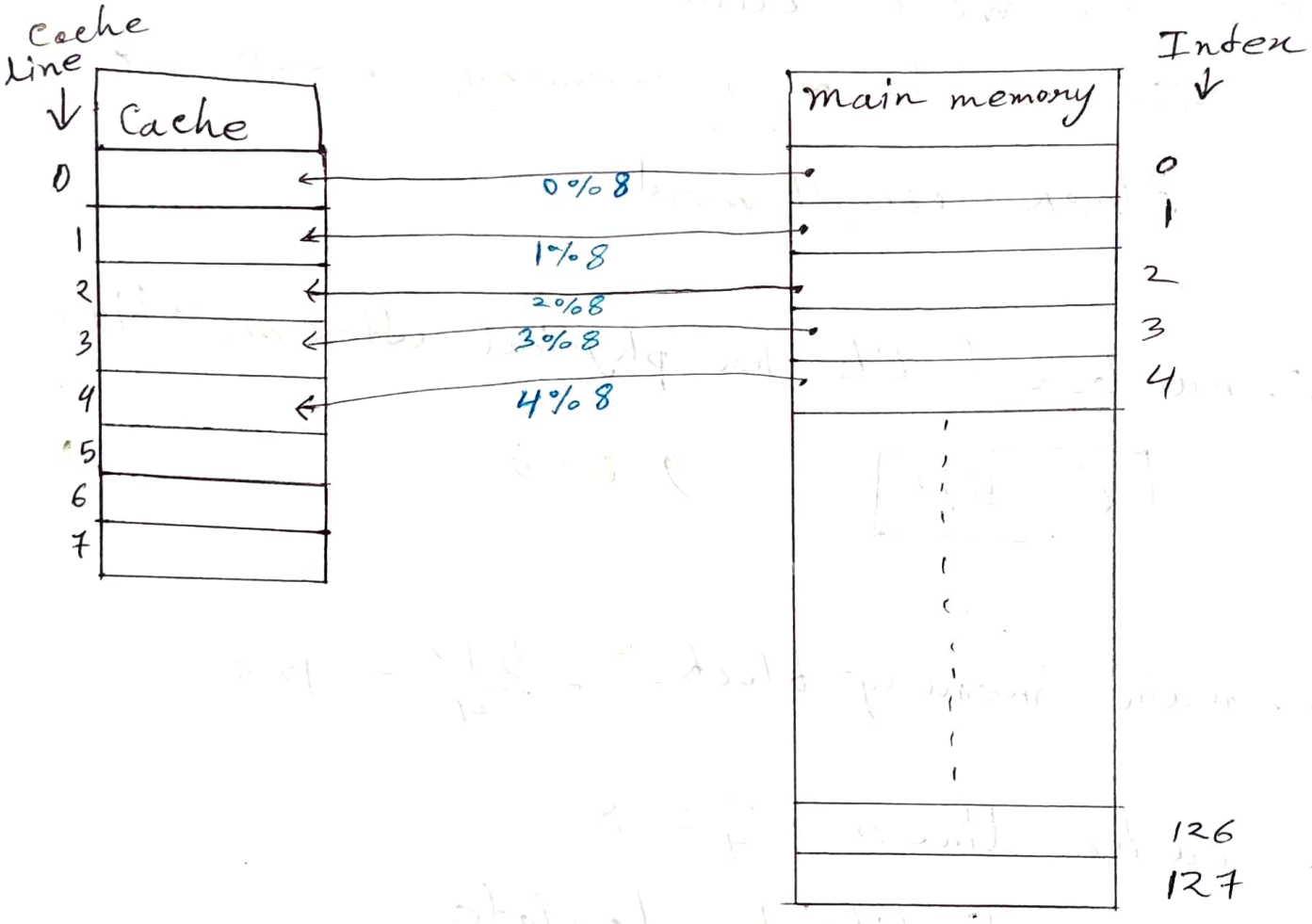
Number of bits to calculate physical address = 9 bits.

Block offset: $4 = 2^2$
∴ 2 bits for block offset.

index bit: $8 = 2^3$
∴ 3 bits for Index.

Tag $= 9 - (3+2) = 4$ bits.

cache configuration for direct mapping:



In direct mapping, main memory location/Data is mapped to exactly one location in the cache.

For mapping, main memory Address is mod by cache line. (% = mod)

Q. 7 Ans:

There are four questions for cache design:

1. Where can a block be placed in the upper level?
2. How is a block found if is in the upper level?
3. Which block should be replaced on a miss?
4. What happens on a write?

Q. 8 Ans:

(i) Direct mapped: Here, R = Replace, A = Access

requested memory block	Hit/Miss	cache lines			
		0	1	2	3
0	miss	main memory block [0]			
8	miss	main memory block [8] (R)			
0	miss	main memory block [0] (R)			
6	miss			main memory block [6]	
6	hit			main memory block [6] (A)	
8	miss	main memory block [8] (R)			
7	miss				main memory block [7]
11	miss				main memory block [11]

(ii) 2-way set associative:

number of sets $= \frac{4}{2} = 2$

here, R = Replace
A = Accen

requested memory block	hit/ miss	cache			
		set 0		set 1	
		0	1	2	3
0	miss	main memory block [0]			
8	miss		main memory block [8]		
0	hit	main memory block [0] A			
6	miss		main memory block [6] R		
6	hit		main memory block [6] A		
8	miss	main memory block [8] R			
7	miss			main memory block [7]	
11	miss				main memory block [11]

(iii) Fully associative:

A = Access
R = Replace

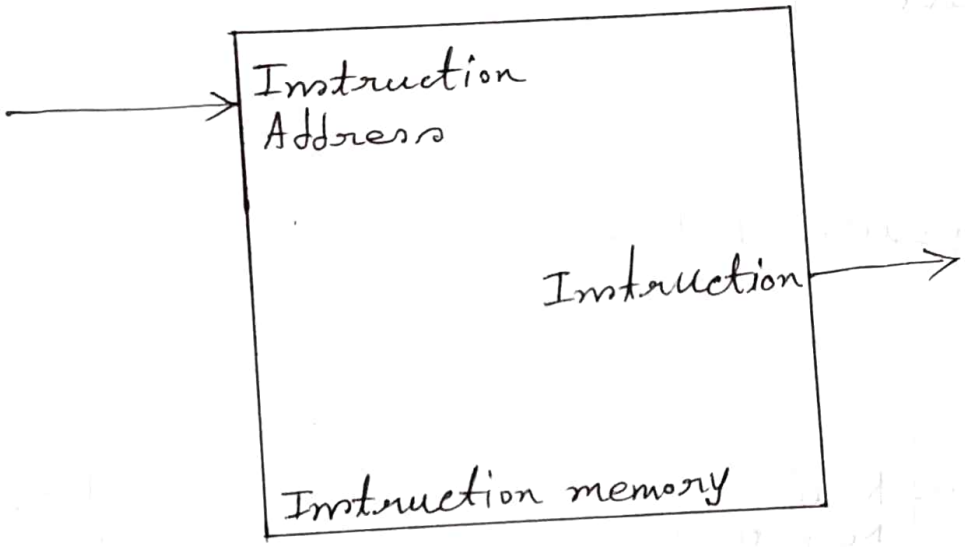
		Cache			
requested memory block	hit/ miss	0	1	2	3
0	miss	main memory block [0]			
8	miss		main memory block [8]		
0	hit	main memory block [0] A			
6	miss			main memory block [6]	
6	hit			main memory block [6] A	
8	hit		main memory block [8] A		
7	miss				main memory block [7]
11	miss	main memory block [11] R			

8.9 Ans:

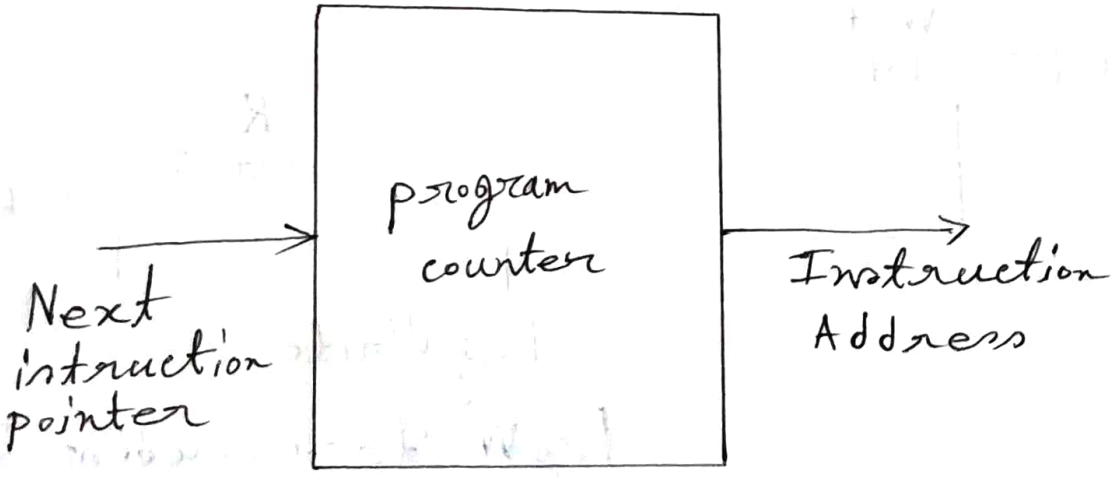
elements of data path:

(i)

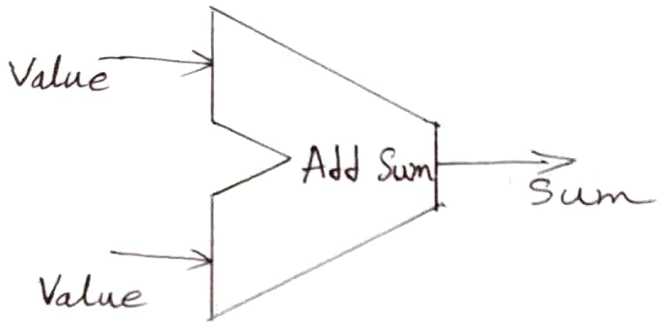
Instruction memory:



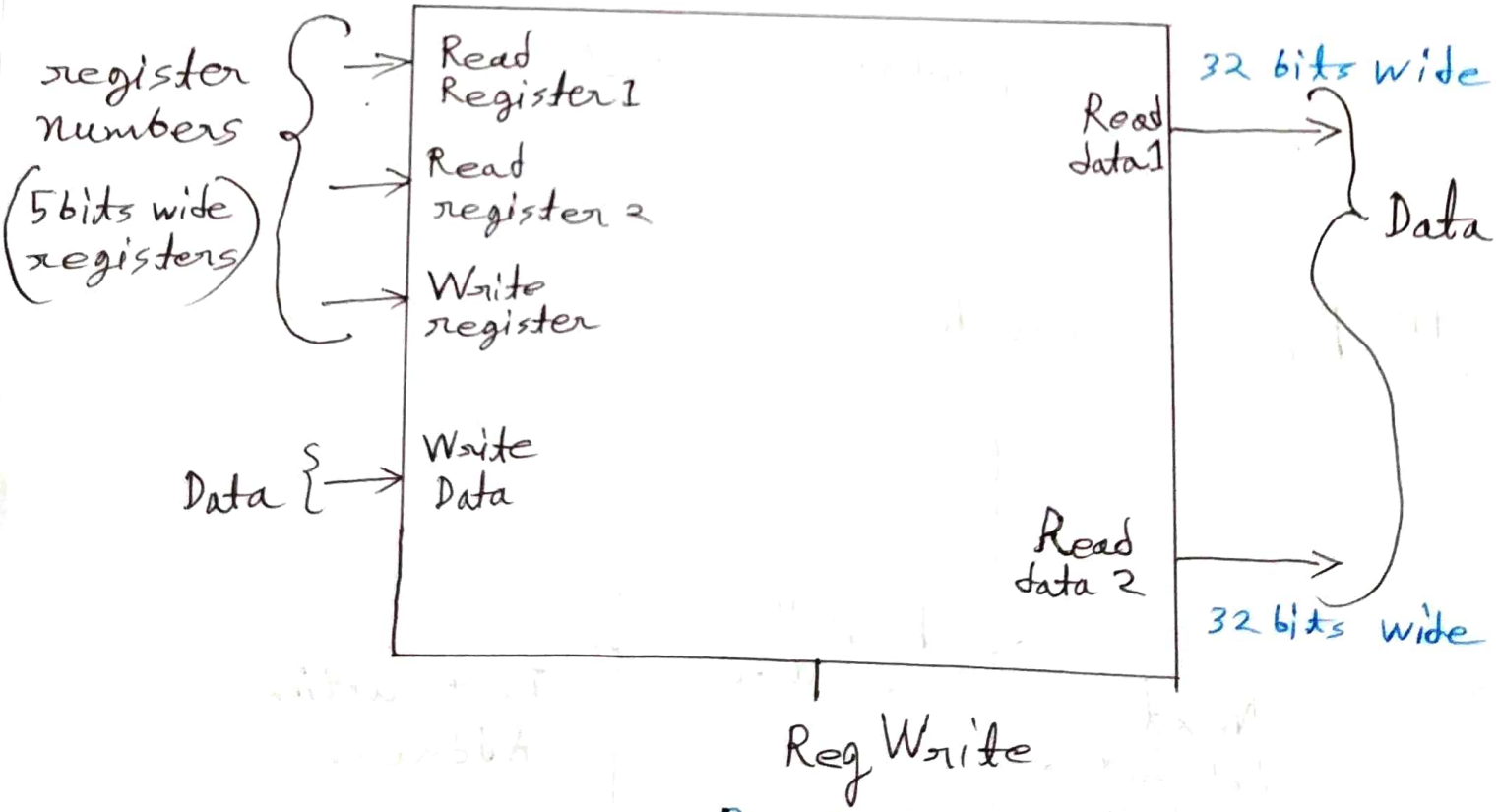
(ii) Program counter:



(III) Adder :



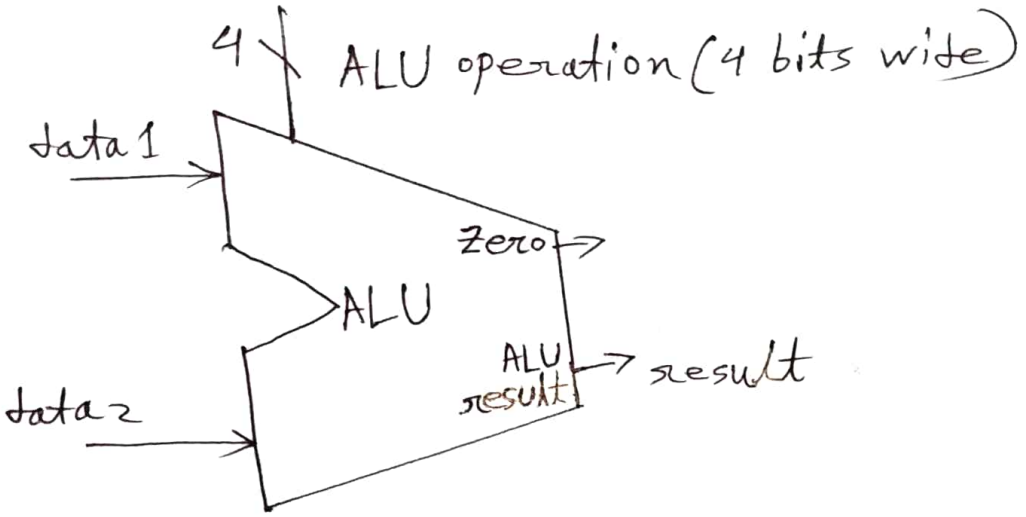
(IV) Register file :



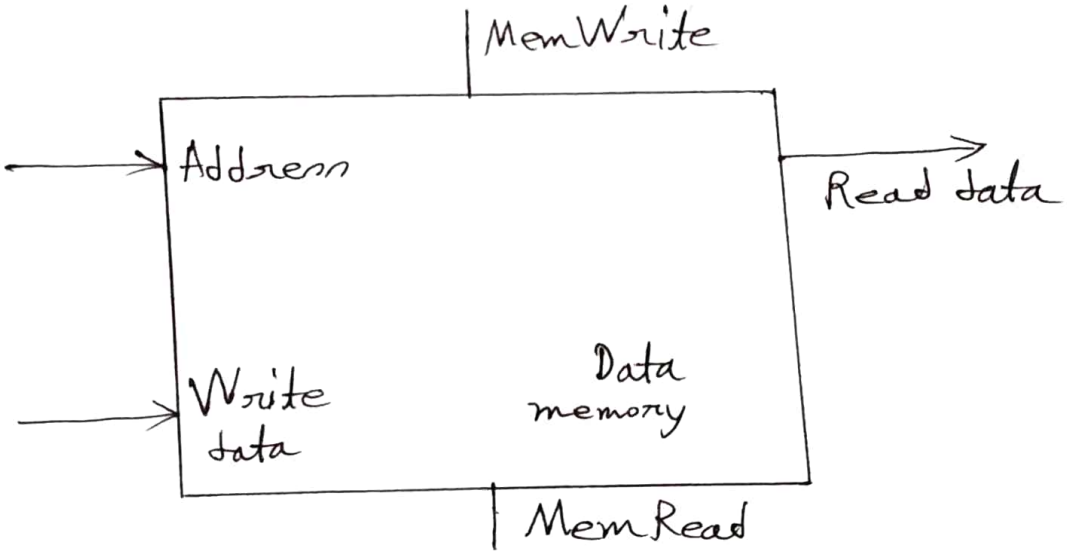
RegWrite = 0, read operation

RegWrite = 1, Write operation

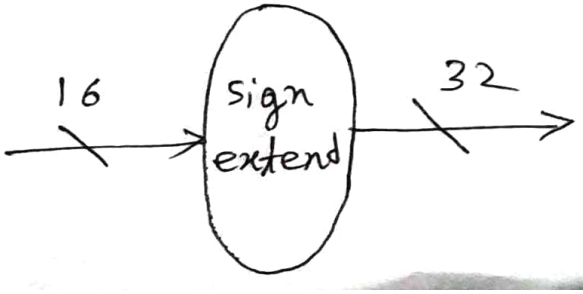
(V) ALU:



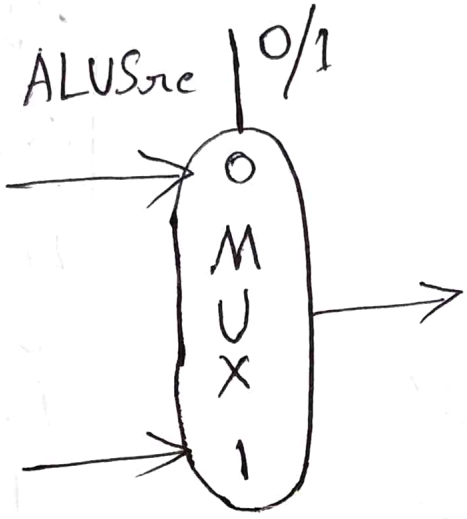
(VI) Data memory:



(VII) Sign extension element takes as input a 16-bit wide value to be extended to 32-bits.



(VIII) Multiplexers (MUX)



Q. 10 Ans:

$$R[i] = R[i+2] - Y$$

$$\therefore R[26] = R[28] - Y$$

My ID: 1720126

$\therefore i = 26$

Let,

Registers,

$R = S_0$

$Y = S_2$

Now,

1. lw $t0, 112($S_0$)
2. Sub $t0, $t0, S_2
3. Sw $t0, 104($S_0$)

Here, Diagrams are drawn by black Ink and before Register file everything are same. So after Instruction memory,

1. Blue Ink used to show the data path for instruction 1 (I-type Instruction)
2. Pink Ink used to show the data path for instruction 2 (R-type Instruction)
3. Purple Int used to show the data path for instruction 3 (I-type Instruction)

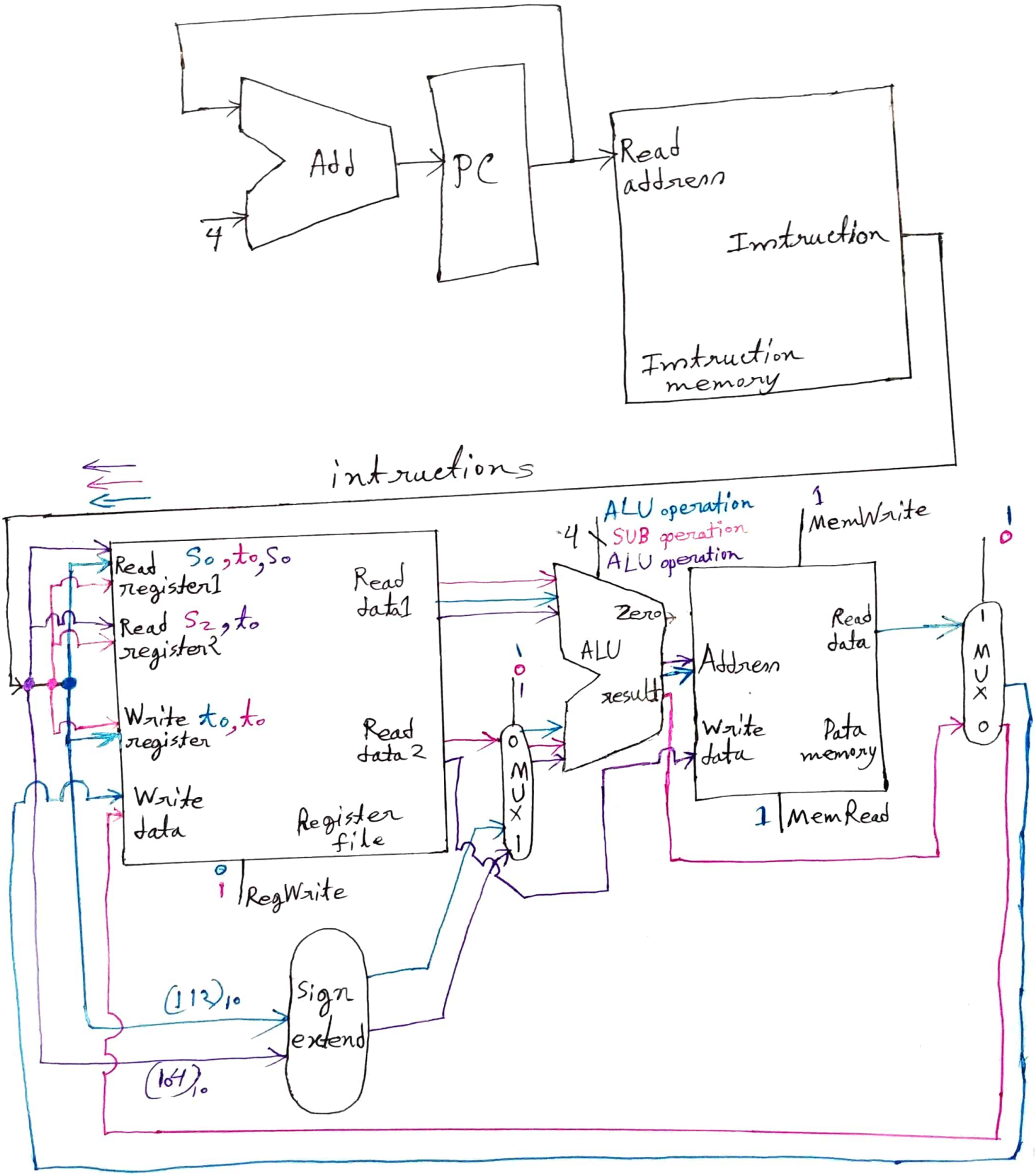


Diagram: Datapath

Steps :

1. At first, the program counter will hold the address of first instruction (lw $t0, 112($s0)).
2. Then, the address will be sent to instruction memory (at Read address).
3. Instruction memory will decode the instruction and result send to register file and sign extend.

Where, s_0 → Read register 1
t_0 → Write register
$(112)_{10}$ offset → sign extend

4. Sign extend will make 16 bits wide offset to 32 bits and send it to MUX, where MUX ALUSrcc is set.
5. 'Read data1' from registers will send the data 's_0' to ALU and the MUX will send the offset of 32 bit to ALU
6. ALU result will send to data memory → address, where MemRead is set.

7. Data memory will send read data to 2nd MUX, where MUX is set.

8. MUX will send the value to
Register file → Write data
Where RegWrite = 0

9. Now, add 4 bytes to the PC value to obtain the word-aligned address of the next instruction (Sub t_0, t_0, s_2).

10. Then the address of the instruction (Sub t_0, t_0, s_2), send to instruction memory → at Read address.

11. Instruction memory will decode the instruction and result send to register file

Where, t_0 → Read register 1
s_2 → Read register 2
t_0 → Write register

12. Register file will send Read data 1 & Read data 2 to ALU and MUX, where MUX is not set.

then the MUX will send the data to ALU

13. Now here ALU is set to SUB operation.

14. Result of ALU will send to 2nd MUX, where it is set.

15. MUX will send the value to Register file to write, when RegWrite is set.

16. Now again add 4 bytes to the Pc value to obtain the address of the next instruction.

17. The address of the instruction (sw $t0, 104($s0)), send to instruction memory.

18. Instruction memory will decode the instruction and result send to register file and sign extend

where, s_0 → read register 1

t_0 → read register 2

(104) offset → Sign extend

19. From read data 1 (Register file), data will send the data to ALU and from read data 2 the data will send to Data memory → Write data

Here, ALU will send data at Address to Data memory.

20. MemWrite is set in data memory.

program executed.