University of Asia Pacific

Department of CSE

Semester Final Examination, Spring 2020

Name: Rashik Rahman

Reg ID: 17201012

Year: 3rd

Semester: 2nd

Course Code: CSE 317

Course Title: Computer Architecture

Date: 2.11.2020

"During Examination and upload time I will not take any help from anyone. I will give my exam all by myself."

University of Asia Pacific

Admit Card

Final-Term Examination of Spring, 2020

Financial Clearance	PAID

Registration No : 17201012
Student Name : Rashik Rahman
Program : Bachelor of Science in Computer Science and Engineering

SI.NO.	COURSE CODE	COURSE TITLE	CR.HR.	EXAM. SCHEDULE
1	CSE 313	Numerical Methods	3.00	
2	CSE 314	Numerical Methods Lab	0.75	
3	CSE 315	Peripheral & Interfacing	3.00	
4	CSE 316	Peripheral & Interfacing Lab	1.50	
5	CSE 317	Computer Architecture	3.00	
6	CSE 319	Computer Networks	3.00	
7	CSE 320	Computer Networks Lab	1.50	
8	CSE 321	Software Engineering	3.00	
9	CSE 322	Software Engineering Lab	0.75	

Total Credit: 19.50

1. Examinees are not allowed to enter the examination hall after 30 minutes of commencement of examination for mid semester examinations and 60 minutes for semester final examinations.

2. No examinees shall be allowed to submit their answer scripts before 50% of the allocated time of examination has elapsed.

3. No examinees would be allowed to go to washroom within the first 60 minutes of final examinations.

4. No student will be allowed to carry any books, bags, extra paper or cellular phone or objectionable items/incriminating paper in the examination hall.
Violators will be subjects to disciplinary action.

This is a system generated Admit Card. No signature is required.

Admit Card Generation Time: 27-Oct-2020 08:43 PM

17201012

Answer to the Q. NO. 1

~~X=12;~~ Here, X = 12

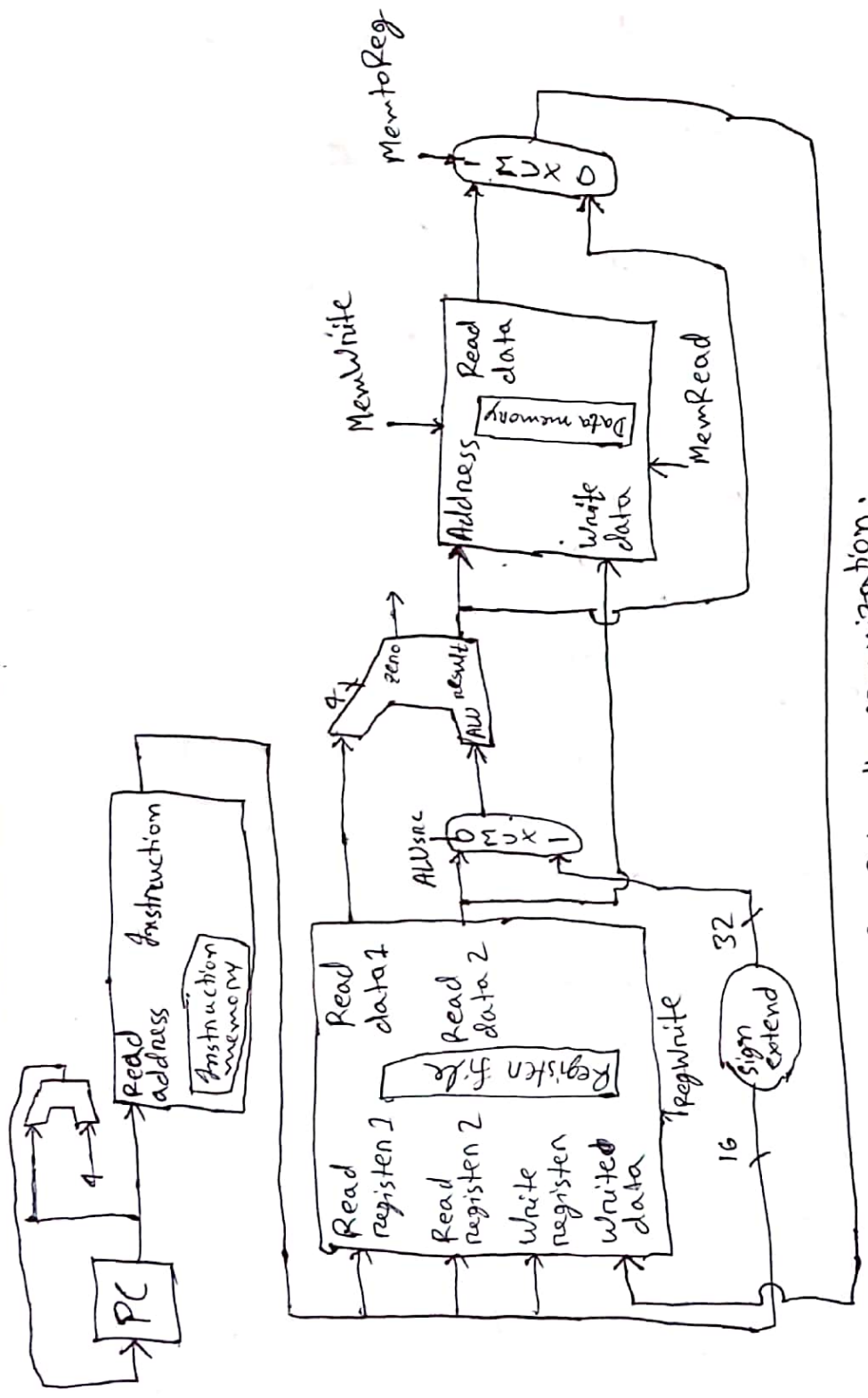


fig: Datapath organization.

Instruction address is placed on the instruction memory from program counter. As each instruction is of 4 words. So every time PC passes an address to instruction memory it increaments by 4 using an adder. Instruction memory retrieves and passes the instruction corresponding to that instruction address.

(i) For the 1st instruction (lw $s4, 12($s0)):

As it is an I-type instruction so the number of $s0 that is 16 would be passed to read register 1 and the content of $s0 that is the base address would come out of read data 1. The ALUsrc MUX is SET to indicate it's an I-type instruction. 16bit offset would be converted to 32bit offset using sign extend. Now this base address and offset would be passed to ALU and we'll get the physical address from ALU result. This physical address is passed to the address port of the data memory. Now MemRead and MemtoReg are SET so data memory will fetch the data corresponding to that physical address, and pass it to write data in Register file. RegWrite is SET and the number of $s4 that is 19 is passed to write register. This is how data is fetched from memory and stored in register.

(ii) For the 2nd instruction (sw $t0, 12($s0)):

As it is an I-type instruction so the ALUsrc is SET. The number of $s0 that is 16 is passed to read register 1 and we get the content of $s0 that is the base address from read data 1. Now the number of $t0 that is 8 is passed to read register 2 and it's content that is the data to be stored is the output of read data 2. The 16 bit offset is converted to

32 bit offset using sign extend. Now the base address and offset is passed to ALU. ALU adds them thus we get the physical address in where we want to store the data. This physical address and data to be stored is passed to data memory where MemWrite is SET so data is written into memory.

(iii) For the 3rd instruction (Add $t_0, $S_1, $S_2):

As it is a R-type instruction so ALUSrc MUX is CLEAR. The number of S_1 & S_2 that is 17 & 18 is passed to read register 1 & read register 2 accordingly. Thus we get the content of S_1 & S_2 from read data 1 and read data 2. These two data goes to ALU, ALU add them. then the result directly goes to MemtoReg MUX as it is CLEAR so it'll pass the data to register file's write data section. The number of t_0 that is 8 is passed to write register, and RegWrite is SET so the data is stored in t_0.

(iv) For the 4th instruction (A[x] = Y + A[x+8]):

$$A[12] = Y + A[12+8] = Y + A[20]$$

Now,

$A \rightarrow \$S_0$ (16)

$Y \rightarrow \$S_1$ (17)

temp $\rightarrow \$t_0$ (8)

So the instructions are: a) lw $t_0, 80($S_0)$

b) add $t_0, $t_0, S_1

c) sw $t_0, 48($S_0)$

(9)

17201012

a) lw $t0, 80($s0)

It is I-type instruction so ALUsrc is SET. The number of $s0 that is 16 is passed to read register 1. And we'll get it's content that is the base address from read data 1. 16 bit offset is converted to 32 bit using sign extend. Now the base address and offset will goto ALU and ALU will add them thus we'll get the physical address from ALU result. This physical address goes to data memory where MemRead & MemtoReg MUX is SET so the data memory fetches data from that address and pass that data to write register section of register file. Number of $t0 that is 8 is passed to write register and RegWrite is SET so the data is saved to $t0.

b) add $t0, $t0, $s1:

The second instruction is R-type so ALU-src is CLEAR. The number of $t0 & $s1 that is 8 & 17 is passed to read register 1 & read register 2. Thus we get the content of these two from read data 1 & read data 2. These two data is passed to ALU. ALU adds them thus we get the result from ALU result. The result is passed to MemtoReg MUX but as it is CLEAR so the MUX passes the result to write data section of register file. The number of $t0 that is 8 is passed to write register and RegWrite is SET. So the result is stored is $t0

(c) sw $t0, 8 48($s0) :

As it is an I-type instruction so ALUsrc is SET. The number of $s0 that is 16 is passed to read register 1 and the number of $t0 that is 8 is passed to read register 2. Thus we get the content of $s0 that is the base address from read data 1 and content of $t0 that is the data to be stored from read data 2. 16 bit offset is converted to 32 bit offset. Base address and offset is passed to ALU thus we get physical address from ALU result. This physical address and data to be stored is passed to data memory. As MemWrite is SET so data memory stores the data ~~at the~~ in that physical address.

(6)
17201012

Answer to the Q. No. 2 (a)

Block addresses:

12+20, 12+16, 12+25

⇒ 32, 28, 37

(i) Direct mapped:

Requested address	Hit/miss	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
32	miss	mem[0]																
28	miss												mem[12]					
37	miss					mem[5]												

(ii) Set associative:

a) 2way:

$\frac{16}{2} = 8$

So we'll mod the addresses with 8 and decide which set the block should mapped to and will use LRU replacement technique.

P.T.O

17201012

		set 0		set 1		set 2		set 3		set 4		set 5		set 6		set 7	
Requested address	Hit/ miss	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
32	miss	mem[0]															
28	miss									mem[8]							
37	miss											mem[37]					

b) 4 way:

$\frac{16}{4}$ = 4 sets.

		set 0				set 1				set 2				set 3			
Requested address	hit/ miss	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
32	miss	mem[0]															
28	miss		mem[7]														
37						mem[4]											

c) 8 way:

$\frac{16}{8}$ = 2 sets

Requested address	miss/ hit	0

(8)

17201012

Requested address	hit/miss	set 0								set 1							
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
32	miss	mem[0]															
28	miss		mem[1]														
37	miss								mem[8]								

d) 16 way/full associtive:

Requested address	hit/miss	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
32	miss	mem[0]															
28	miss		mem[1]														
37	miss			mem[2]													

17201012

Answer to the Q. No. 2(b)

(i) Given,

Cache size = 32 words

Main memory size = 512 words

Block size = 4 words,

∴ Memory blocks = $\frac{512}{4}$ = 128 blocks

Cache lines = $\frac{32}{4}$ = 8 blocks.

Total bit for physical address = $\log_2(512)$ = 9 bits

Index = $\log_2$ (cache lines) = $\log_2(8)$ = 3 bit

Offset = $\log_2$ (block size) = $\log_2(4)$ = 2 bit

Tag bit = 9 − 3 − 2 = 4 bit Ans.

(ii) Given,

Cache size = 16 words

Main memory size = 64 words

block size = 4 words

∴ Memory blocks = 64/4 = 16 blocks

Cache line = 16/4 = 4 blocks.

∴ Total bit = $\log_2(64)$ = 6 bits

Index = $\log_2(4)$ = 2 bits

Offset = $\log_2(4)$ = 2 bit.

Tag bit = 6 − 2 − 2 = 2 bit Ans.

(10)

17201012

(iii) Given,

cache size = 16 words

main memory size = 128 words

block size = 4 words

∴ memory block = 128/4 = 32 blocks

cache line = 16/4 = 4 blocks

∴ Total bits = $\log_2(128)$ = 7 bits

Index = $\log_2(4)$ = 2 bit

Offset = $\log_2(4)$ = 2 bit

Tag bit = 7 - 2 - 2 = 3 bit

Ans.

(iv) Given,

cache size = 32 words

memory size = 1024 words

block size = 4 words

∴ ~~Total bits = $\log_2(1024)$ =~~

memory block = 1024/4 = 256 blocks

cache line = 32/4 = 8 blocks

∴ ~~Index = $\log_2(1024)$ =~~

Total bits = $\log_2(1024)$ = 10 bits

∴ Index = $\log_2(8)$ = 3 bit

∴ Offset = $\log_2(4)$ = 2 bit

∴ Tag bit = 10 - 3 - 2 = 5 bit

Ans

(v) Given,

Cache size = 8 words
memory size = 16 words.
block size = 4 words

memory block = 16/4 = 4 blocks
cache line = 8/4 = 2 blocks

∴ Total bit = $\log_2(16)$ = 4 bit
∴ Index = $\log_2(2)$ = 1 bit
∴ offset = $\log_2(4)$ = 2 bit
∴ Tag bit = 4 - 1 - 2 = 1 bit

Ans.

Answer to the Q.No. 3(a):

Instructions = 12+7 = 19
Stage = 5

(12)
1720102

S_1	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}	I_{18}	I_{19}				
S_2		I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}	I_{18}	I_{19}			
S_3			I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}	I_{18}	I_{19}			
S_4				I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}	I_{18}	I_{19}		
S_5					I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}	I_{18}	I_{19}	

Here $k=5$, $n=19$:

Total time for pipeline $= k+n-1 = 5+19-1 = 23$ clock cycles

Total time for non pipeline $= n \times k = 5 \times 19 = 95$ clock cycles.

$$\text{Speed up} = \frac{\text{Total clock cycle for non pipeline}}{\text{Total clock cycle for pipeline}} = \frac{95}{23} = 4.1304$$

$$\text{Utilization} = \frac{\text{Used blocks}}{\text{Total blocks}} = \frac{5 \times 19}{(5+19-1)\times 19} = \frac{95}{95+20} = \frac{95}{115} = 0.8261 = 82.61\%$$

Ans

Answer to the Q No: 3(b)

Non pipeline:

Instruction latency = Σ(stage time)

= (12 + 15 + 8 + 18 + 20) ns

= 73 ns

instructions = 1000 + 12 = 1012

∴ execution time = 73 × 1012 = 73876 ns

For pipeline:

Instruction latency = max. stage time × no. of stages

= 20 × 5 = 100 ns

execution time = 100 + 20 × 1011

= 20320 ns.

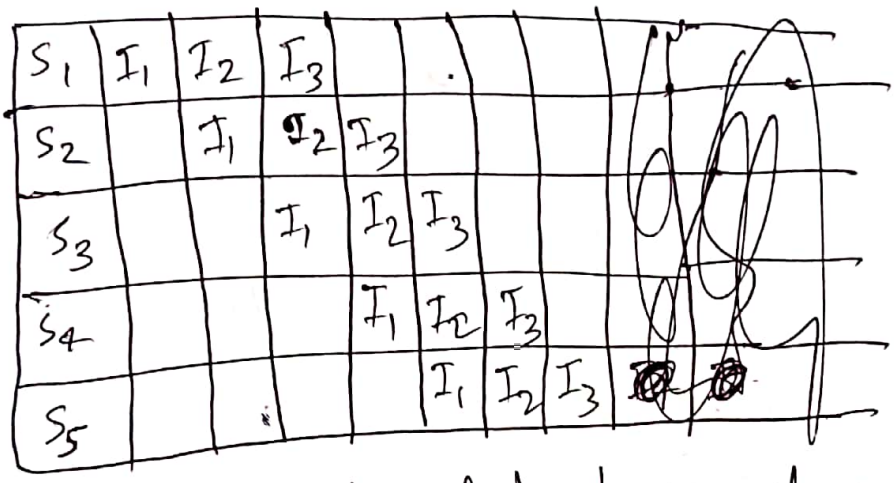
$$\therefore \text{Speed up} = \frac{\text{Non-pipeline time}}{\text{Pipe line time}} = \frac{73876}{20320} = 3.6356$$

Ans.

(14)
17201012

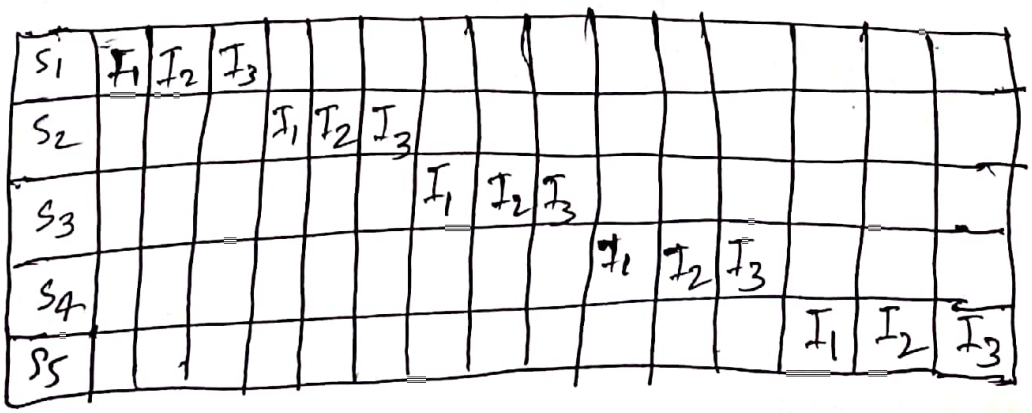
Answer to the Q. No. 3(c)

For pipeline;



There would be data hazard as the S_2 of instruction 2 is loaded to pipeline before instruction 1 stores new value to S_2, same goes for instruction 3 it'll also cause data hazard.

For non-pipelines



17201012

Answer to the Q. No. 9(a) (OR)

Four questions:

i) Where can a block be placed in upper level?

ii) How a block is found if it is in upper level?

iii) Which block should be replaced on a miss?

iv) What happens on a write?

To identify a block we devide block address into two segments block number and block offset. Using block number and offset a word can be identified in a block. Say to identify word 5. So 6 bit of 5 is 000101. Block number 0001 and offset 01 so word 5 is in block 1 and offse 1.

(16)
17201012

Answer to the Q.100.4(b) OR

$x[12] = z + x[12+11] - w$

$x[12] = z + x[23] - w$

Instructions:

~~i) lw $t0,~~

$x \rightarrow \$s_0$ (16)

$z \Rightarrow \$s_1$ (17)

$w \rightarrow \$s_2$ (18)

temp → ~~8~~ $t0 (8)

i) lw $t0, 92($s0)

ii) add $t0, $t0, $s1

iii) sub $t0, $t0, $s2

iv) sw $t0, 48($s0)

Machine code:

(i)

100011	10000	01000	0000 0000 0101 1100
op	rs	rt	offset

(ii)

000000	01000	10001	01000	00000	100000
op	rs	rt	rd	shamt	funct

(iii)

000001	01000	10010	01000	00000	100010
op	rs	rt	rd	shamt	funct

(iv)

101011	10000	01000	0000 0000 00110000
OP	rs	rt	offset

Answer to the Q.No. 7 (c) OR

In computer architecture memory hierarchy separates computer storage into a hierarchy based on response time. Since response time, complexity & capacity are related so levels may also be distinguished by their performance. The Goals of memory hierarchy are keep information close to ALU, increase speed of processing etc.