University of Asia Pacific

Department of CSE

Mid-Semester Examination, Spring 2020

Name: Rashik Rahman

Reg ID: 17201012

Year: 3rd

Semester: 2nd

Course Code: CSE 317

Course Title: Computer Architecture

Date: 25.08.2020

17201012

Answer to the Q. No. 1(a)

There are 5 classic components of a computer.

→ Input
→ Output
→ Memory
→ Datapath
→ Control.

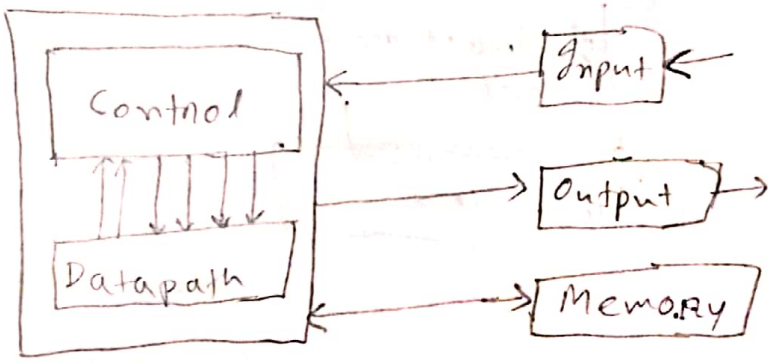


fig: basic components.

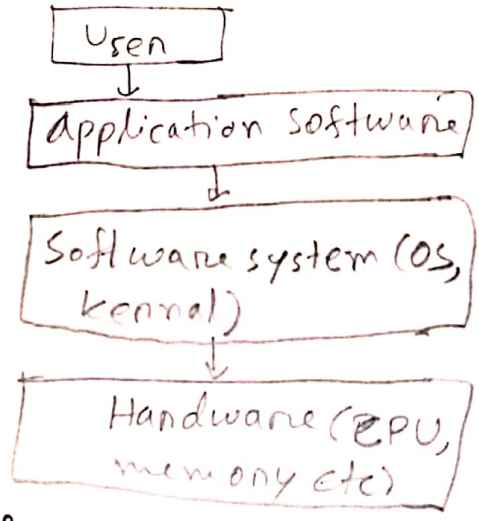
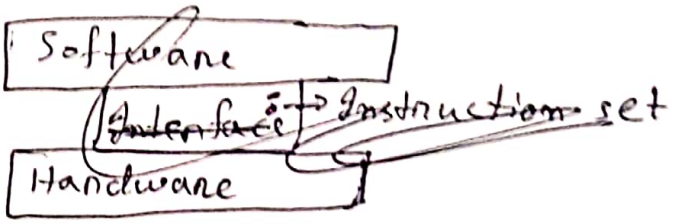
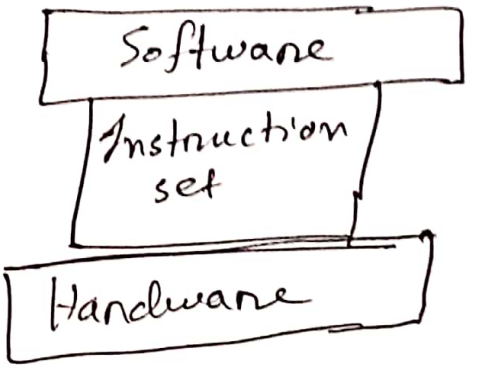


fig: Layer of a computer

(2)
17201012



Instruction set is the interface between handware and low level software.



Answer to the Q.NO.1 (b)

Let, I = no. of instructions

execution time of A, $E_A = I \times 1.8 \times 300ns$

$= I \times 540\ ns$

execution time of B, $E_B = I \times 1.4 \times 550ns$

$= I \times 770\ ns$

We got these by using performance equation 2;

execution time = number of instructions × CPI × clock cycle time.

So as E_A is less then E_B then we can say computer A is fasten.

We know, $P_x = \frac{1}{E_x}$

$$\therefore \frac{Performance_A}{Performance_B} = \frac{E_B}{E_A} = \frac{I \times 540ns}{I \times 770ns} = 0.70$$

$\therefore Performance_A = 0.70 \times$ Performance of B.

So computer A is 0.70 times fasten then computer B.

(4)
17201012

Answer to the Q. NO. 2(a)

In MIPS architecture there are 3 types of instruction class. Those are,

→ R-type
→ I-type
→ J-type

R-type instruction is faster than I-type instruction. Cause, R-type instruction is for arithmetic operation. Here the source and destination operand's value are stored in registers. And I-type is generally used of memory operation. Like loading data from memory or storing data to memory. R-type works with register and I-type mostly works with memory.

So as R-type is register reference instruction and I-type is memory reference instruction that's why R-type is faster.

Answer to the Q. No. 2(b)

$X[i] = Z + X[i+5]$

Here, $i = 12$

So, $X[12] = Z + X[17]$

let,

$X = S_0$

$Z = S_1$

So,

$S_0 = 16 = (10000)_2$

$S_1 = 17 = (10001)_2$

$add = 32 = (100000)_2$

$lw = 33 = (100001)_2$

$sw = 34 = (100010)_2$

$to = 8 = (01000)_2$

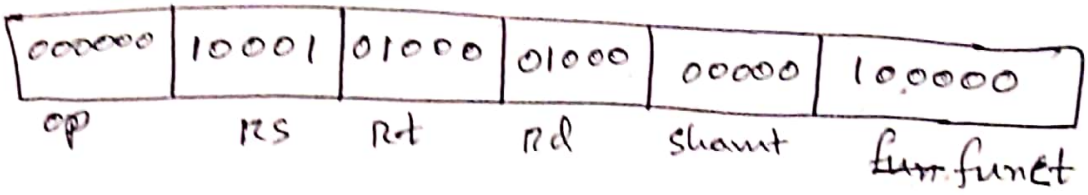
Machine code:

i) lw $to, 68($S0)

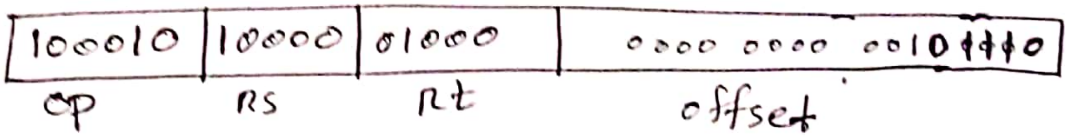
100001	10000	01000	0000 0000 0100 0100
op	rs	rt	offset

(6)
17201012

ii) add $t0, $s1, $t0



iii) SW $t0, 48($s0)



Answer to the Q. No. 3 (a)

Here,

multiplicand = (2's 5) + 4

$\therefore m = 2+4 = 6 = 00110$

$\therefore -m = 11010$

multiplier = $-4 = 11100$

Iterations	Step	Multiplicand	Product	Extra bit
0	Initial	00110	00000 11100	0
1	1. P₀ Ex = 00 1c. No operation	00110	00000 11100	0
	2. $\vec{P}^1$ ARS	00110	10000 01110	0
2	1. P₀ Ex = 00 1c. No operation	00110	10000 01110	0
	2. $\vec{P}^1$ ARS	00110	11000 00111	0
3	1. P₀ Ex = 10 1b. P(L) = P(L) − M	00110	00010 00111	0
	2. $\vec{P}^1$ ARS	00110	10001 00011	1
4	1. P₀ Ex = 11 1c. No operation	00110	10001 00011	1
	2. $\vec{P}^1$ ARS	00110	11000 10001	1
5	1. P₀ Ex = 11 1c. No operation	00110	11000 10001	1
	2. $\vec{P}^1$ ARS	00110	11100 01000	1
			result = 01000	

$\therefore$ Ans = $-24 = 01000$

(8)

17201012

Answer to the Q. No. 3(b)

Booth's algorithm is based on multiplication version 3. But has slight changes. Like, in booths we do arithmetic right shift and also can add negative multiplicand to P(L). Thus hardware solution would be more like version 3. But just arithmetic right shift.

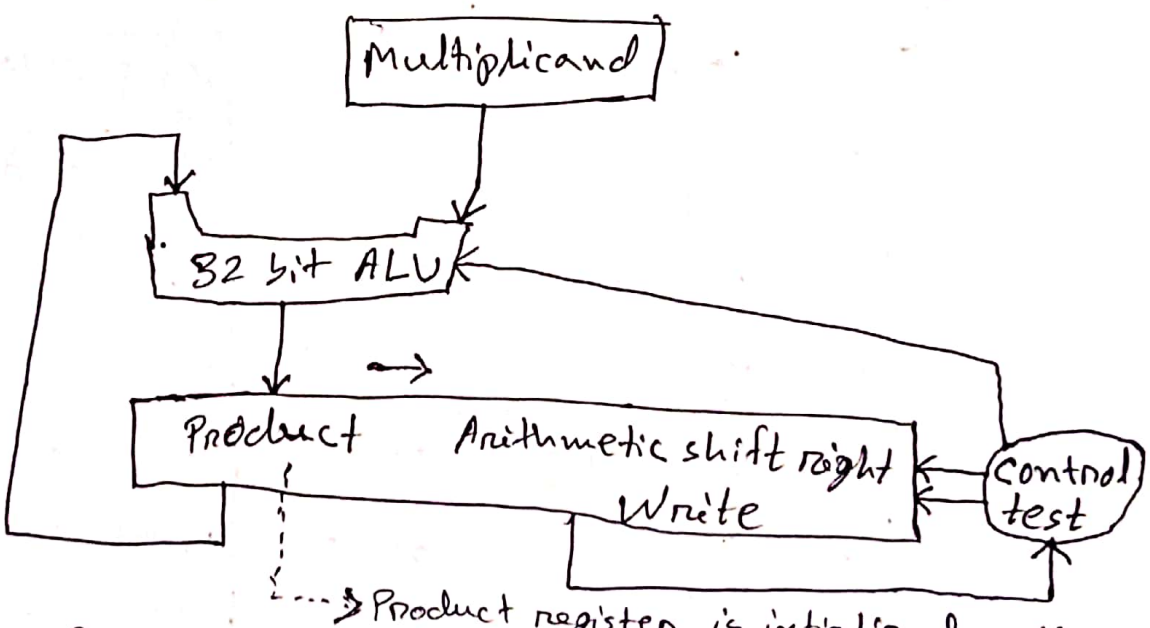


fig: Hardware Solution.