Name: Akib Zabed Ifti ID: 2010113
Section: 07

340 Assignment 1

1

In our code, we have have multiple lines in it and they containing multiple operations. When these codes converted into mips instructions, the cpu needs to keeps track ~~about~~ which line of instruction is executing. And for this task cpu has a register called program counter. The program counter holds the current intruction address. Again, $0 or $zero register is actually the constant '0'. It can only hold 0 in it and for this it not be overcoritten. we use $zero register in different instructions, For example, we can one register value into another register by simply add the $zero register with that register which value we want to move. ~~$so~~ (add $s1, $s0, $zero).

we know that, main ~~more~~ memory contains slots of 8 bits, so for sequential instruction execution

for (64)

$\frac{64}{8} = 8$ will be the increament

and for (128)

$\frac{128}{8} = 16$ will be the increament.

question 2

Instruction given : lw $4, X($5)

we know,

lw , $register , increament x offset($base)

∴ increment for 256 architecture, $\frac{256}{8} = 32$

∴ offset in ~~A[0][5]~~ A[5] is = 5

∴ X = 32 x 5

= 160 .

(Ans)

Question 3

A is in $S0

f is in $S2

i is in $S1 ∴ f = A[i]

MIPS

```
add  $t0 , $s0 , $s1
lb   $s2 , 0($t0)
```

Question 4

X is in $S0, Y is in $S1, Arr is in $S4

i) X = 15Y − 5

For 15Y, ⇒ sll $t0, $s1, 4 [R type]

000000	00000	$s1, $17 10001	$t0, $8 01000	00100	xxxxxx
op	rs	rt	rd	shamt	function

ii) Sub $t0, $t0, $s1 [R type]

000000	$t0, $8 01000	$s1, $17 10001	$t0, $8 01000	00000	xxxxxx
op	rs	rt	rd	shamt	function

• addi $s0, $t0, −5 [I type]

xxxxxx	$t0, $8 01000	$s0, $16 10000	−5 1111111111111011
op	rs	rt	offset

Again, (11) Arr[5] = 2x + Arr[10]

for 2x, add $t0, $s0, $s0 [R type]

000000	$s0, $16 10000	$s0, $16 10000	$t0, $8 01000	00000	xxxxxx
op	rs	rt	rd	shamt	function

Now, Arr[10] ⇒ lw $t1, 40($s4) [I type]

xxxxxx	$s4, $20 10100	$t1, $9 01001	40 0000 0000 0010 1000
op	rs	rt	offset

Now, add $t2, $t0, $t1 [R type]
[2x + Arr[10]]

000000	$t0, $8 01000	$t1, $9 01001	$t2, $10 01010	00000	XXXXXX
op	rs	rt	rd	shamt	func

for, Arr[5] = 2x + Arr[10]

sw $t2, 5x4 [$s4]
⇒ sw $t2, 20($s4) [I type]

XXXXXX	$s4, $20 10100	$t0, $8 01000	20 0000000000010100
op	rs		offset

Question 5

Given Instruction, beq $9, $8, 124

Here the offset is 124

16 bit binary form of 124 = 0000000001111100

Now we have to left shift 2bit of that binary value of 124 or 00000000011111100

affter 2bit lef shift : 00000001111110000

Now extend the sign into 32 bit : 0.....0111110000

PE holds : 0x1278A4B1

In binary : ~~0001 0010 0011~~

0001 0010 0111 1000 1010 0100 1011 0001

Now adding 4 or 100 with the PE number

```
  0001 0010 0111 1000 ..... 0001
(+) 000 - - - - - - - - - - 0100
-------------------------------
  0001 0010 0111 1000 1010 0100 1011 0101
```

Now we have to add this with that 32 bit offset value again,

```
  0001 0010 0111 1000 1010 0100 1011 0101
(+) 0000 0000 0000 0000 0000 0001 1111 0000
-------------------------------------------
  0001 0010 0111 1000 1010 0110 1010 0101
```

∴ branch destination address of that instruction :

0x 1278A6A5

(Ans)

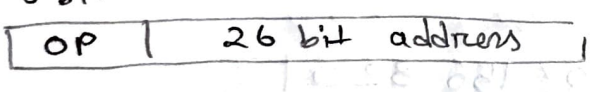
Ques 6

Given instruction : J 1590

PC holds : 0x 00AB1203

binary : 0000 0000 1010 1011 0001 0010 0000 0011

we know the machine format of J type



So the address we have = 1590

26 bit binary form = 00..... 11000110110

26 bit

After 2bit left shift : 000..... 11000110110000

28 bit

Now the PC MSB 4 bit will copy and add on the MSB of that 28 bit address.

∴ MSB 4 bit of PC : 0000

Now after adding this to the 28 bit address we get :

~~0000 0000 0000 0000 0000 0110 0011 0110~~

0000 0000 0000 0000 0001 1000 1101 1000

∴ Hexa decimal form: 0x 0000 18D 8

(Ans)

Question 7

Given Instruction : lw $8, 52($17)

base address : 0x 156 32017

binary form : 0001 0101 0110 0011 0010 0000 0001 0111

32 bit binary form of 52 is:

0000 0000 0000 0000 0000 0000 0011 0100

∴ The memory address of the data that will be loaded in $8 is,

```
  0001  0101  0110  0011  0010  0000  0001  0111
+ 0000  0000  0000  0000  0000  0000  0011  0100
-------------------------------------------------
  0001  0101  0110  0011  0010  0000  0100  1011
```

∴ In Hexadecimal : 0x 15632048

(Ans)

Ques no 8

- MIPS :

```
add $S3, $S3, $zero   →i=0
Loop:
  slti $t0, $S3, 10   →i<10
  beq $t0, $zero, Exit
  sll $t1, $S3, 2
  add $t1, $S1, $t1
  lw $t2, 0($t1)      [loaded A[i]]
```

A → $S1
B → $S2
i → $S3
5 → $S4
1 → $S5

```
    beq $t2, $s4, Else
    sll $t1, $s3, 2
    add $t1, $t1, $s2
    lw  $t2, 0($t1)     [loaded B[i]]
    sll $t1, $t2, 2
    add $t1, $s1, $t1
    lw  $t2, 0($t1)     [loaded A[B[i]]]
    add $t2, $t2, $s5
    add $s3, $s3, $s5   [i++]
    j Loop

Else:
    add $t1, $s3, $s5
    sll $t1, $t1, 2
    add $t1, $s2, $t1
    lw  $t3, 0($t1)     [loaded B[i+1]]
    sw  $t3, 0($t2)     [stored in $t2 = A[i]]
    add $s3, $s3, $s5   [i++]
    j Loop

Exit:
```

Ques 9

MIPS code:

~~$s1~~

```
addi  $s1 , $zero , 20
addi  $s2 , $s1 , -10
addi  $s0 , $zero , 7
add   $s3 , $s2 , $s0

jal sum

sum:
    addi $sp , $sp , -4
    sw   $s0 , 0($sp)
    add  $t0 , $a0 , $a1
    add  $s0 , $t0 , $a2
    add  $v0 , $s0 , $zero
    lw   $s0 , 0($sp)
    addi $sp , $sp , 4

    jr ra
```

Given,

a → $s0
x → $s1
y → $s2
z → $s3

→ [Stored value of s0 in stack]

[x = a0
  y = a1
  z = a2]

→ [retrieve the value of $s0 from stack]