Name: Md Raihan Islam Bhuiyan

Id : 2020239

Section: 12

Course : CSE 340

Assignment: 1

Ans to the Q: N: 1

Program Counter: A program counter is a register in a computer processor that contains the adress of the instruction being executed at the current time. After each instruction's execution, the program counter increases its stored value by 1. This makes the computer to work sequentially.

Register $0: Register $0 always holds the constant 0. ~~This is~~ the value 0 is so useful that there $0

register was created. for example,

int a = 10. If we convert it to MIPS code, it will look like ~~the~~ below.

addi $s1, $0, 10.

Here, we used register $0.

In memory adress, each slot contains 8 bits. So, for 64 bit architecture, the increment in memory adress is, $64 \div 8 = 8$. For 128 bit architecture, it is $128 \div 8 = 16$.

Ans to the Q: N: 2

lw, $4 , X ($5)

Here, x is the offset. We have to get the value from A[5].

As the index number is 5, and it is a 128 bit structure,

$$X = \frac{128}{8} \times 5 = 80$$

We multiplied 16 with the index because there are 128 bits = 16 bytes in a word. Every index has 16 bytes and to reach index 5, we need to multiply 16 with 5.

Ans to the Q:N:3

Given instruction,

$X = 3Y + 5Z + 10$

X → $s0
Y → $s1
Z → $s2
Arr → $s4

sll $t0, $s1, 1 # shift left by 1 bits

$t0 = 2Y

R-type instruction (shift left)

000000	00000	10001	01000	00001	xxxxxx
OP(6)	rs(5)	rt(5)	rd(5)	shamt (5)	funct (6)

Machine code

add $t0 , $t0, $s1

#t0 = 2Y + Y
= 3Y

$t0 = 8 = 01000

$s1 = 17 = 10001

add is an R type instruction

000000	01000	10001	01000	00000	xxxxxx
OP(6)	rs(5)	rt(5)	rd(5)	shamt(5)	funct(6)

Machine Code

sll $t1, $s2, 2

$t1 = 9 = 01001

$s2 = 10010

$12_{10} = (00010)_2$

shift left 2 by 2 bits

2 ← $s2

$t1 = 42

shift left is an R type instruction

000000	000000	10010	01001	00010	xxxxxx
op (6)	rs(5)	rt(5)	rd (5)	shamt(5)	funct (6)

Machine Code

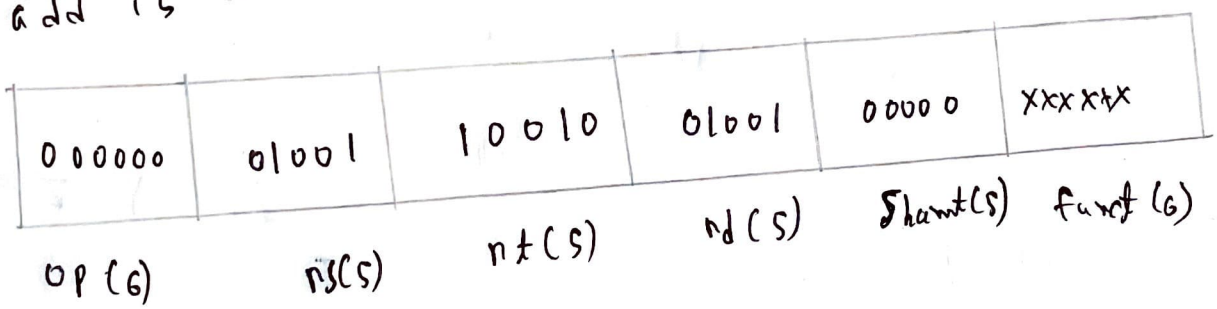
add $t1, $t1, $s2

$t1 = 4 + 2
= 52

$t1 = 9 = 01001

$s2 = 10010

- add is an R type instruction



Machine Code

add $t2, $t0, $t1 | $t2 = 3Y+5Z

$t2 = 01010

add is an R type instruction.

000000	01000	01001	01010	00000	XXXXXX
op(6)	rs(5)	rt(5)	rd(5)	shamt(5)	funct(6)

addi $s0, $t2, 10 $0 = 3Y+5Z+10

addi is an I type instruction

	$t2	$s0	
XXXXXX	01010	10000	0000000000001010
op(6)	rs(5)	rt(5)	constant(16)

Again,

$Y = 4X + Arr[10] + 22$

$x \leftarrow \$s0$

$y \leftarrow \$s1$

$z \leftarrow \$s2$

$Arr \leftarrow \$s4$

MIPS Code:

$\$t0 = 4X$

```
sll $t0, $s0, 2
```

sll is an R-ty instruction.

000000	00000	10000	01000	00010	xxxxxx
-OP(6)	rs(5)	rt(5)	rd(5)	shamt(5)	funct(6)

MIPS Code:

lw $t1, 40($s4)

$t1 = Arr[10]

load is an I type instruction.

$s4 = $(20)_{10} = (10100)_2$

xxxxxx	10100	01001	0000000000101000
op(6)	$s4 rs(5)	$t1 rt(5)	16 bit offset

MIPS Code

sll $t2, $s2, 1

$t2 = 22

shift left is an R type instruction.

		$s2	$t2	shift amount	
000000	000000	10010	01010	00001	xxxxxx
op(6)	rs(5)	rt(5)	rd(5)	shamt(5)	funct(6)

MIPS CODE:

add $t3, $t0, $t1 $t3 = 4x+Arr[10]

add is an R-type instruction.

$t3 = 01011

	$t0	$t1	$t3		
000000	01000	01001	01011	0000 0	xxxxxx
op(6)	rs(5)	rt(5)	rd(5)	shamt(5)	funct(6)

MIPS CODE:

add $s1, $t3, $t2 $s1 = 4X+Arr[10]+22

RTypy instruction.

	$t3	$t2	$s1		
000000	01011	01010	10001	00000	xxxxxx
op(6)	rs(5)	rd(5)	rd(5)	shamt(5)	funct(6)

Now,

Instruction is,

$Z = 5X + 5Y + Z$

MIPS CODE:

$\$t0 = 4X$

```
sll $t0, $s0, 2
```

R type instruction.

		$s0	$t0	2	
000000	000000	10000	01000	00010	xxxx xx
op	rs	rt	rd	shamt	funct

MIPS CODE:

$\$t0 = 5X$

```
add $t0, $t0, $s0
```

R type instruction.

000000	01000	10000	01000	00000	xxxx xx
	$t0	$s0	$t0		

MIPS CODE:

$t1 = 4*

sll $t1, $s1, 2

R-type instruction.

op	rs	rt	rd	shamt	funct
000000	00000	10001	01001	00010	xxxxxx
	$s1	$t1	2		

MIPS CODE:

$t1 = 5*

sll $t2, $t1, $s1

R-type instruction:

op	rs	rt	rd	shamt	funct
000000	01001	10001	01001	00000	xxxxxx
	$t1	$s1	$t1		

MIPS CODE:

add $t2, $t0, $t1 $t2 = 5x + 5y

R type instruction.

$t0 $t1 $t2

000000	01000	01001	01010	000000	xxxxxx
op	rs	rt	rd	shamt	funct

$s2 = 5x + 5y + z

MIPS CODE:

add $s2, $t2, $s2

000000	01010	10010	10010	00000	xxxxxx
op	rs	rt	rd	shamt	funct

Again,

Instruction is,

Arr[10] = X + Y - 30

MIPS CODE:

$t0 = X + Y

```
add $t0, $s0, $s1
```

	$s0	$s1	$t0		
000000	10000	10001	01000	00000	xxxxxx
op	rs	rt	rd	shamt	funct

MIPS CODE:

```
addi $t1, $t0, -30
```

-30 = 1111111111100010
I type instruction

xoxxxx	01000	01001	1111111111100010
	$t0	$t1	16 bit integer

MIPS CODE:

40($s4) = x + y - 30

sw $t1, 40($s4)

xxxxxx	01010	01001	0000000000101000
op	rs	rt	constant

Answer to Q.No: 4

Given that,

B is in $s0

i is in $s1

f is in $s2

Statement → f = B[i]

MIPS CODE:

```
SLL   $t1, $s1, 2
add   $t2, $s0, $t1
lw    $s2, 0($t2)
```

Here, the index number is i. For 32 bit architecture, we multiply the index number with 4 ~~. That is~~ and add it to the base address of the array. That is why we left shifted i 2 bits. and then added it with the address of

B.

Answer to the Q:N:S

Given instruction,

int a = 100

Let, a = $s0

MIPS CODE:

addi $s0, $0, 100

We have used zero register. When we add something with $0, the value of it does not change. So, we added 100 with $0 and kept it in $0 (a).

Ans to the Q.N: 6

Given Date is, AED 1008B. This is a Hexadecimal Number and it is a 32 bit Number when we convert it to Binary. Every digit contains a 4 digit binary number.

In memory, each slot contains 8 it data or 1 byte data. So, we need 4 slots to store this in memory.

There are two orders to store the data in memory. They are:

(i) Big Indian order.

(ii) Little Indian order.

Big Indian Order:

When we follow Big Indian Order, the MSB is stored in the least memory address and LSB is stored in the highest memory address.