Name : Tanjim Reza

Student ID : 20101065

Section : 12

Course : CSE 340

20101065_TanjimReza_12.PDF

Ans: to the que no: 01

There are differences between Program counter and Register $0. The first difference is PC or Program Counter is not fixed, it is a instruction pointer which gets changed. On the other $0 ($zero) register is fixed, it cannot be changed. There are some other things like $zero can be used to swap values between two registers. Without exceptions PC has sequential execution.

Now, for 64bit architecture the increment in memory address for sequential instruction will be $\frac{64}{8} = 8$ and

for 128-bit that would be $\frac{128}{8} = 16$

Ans: to the que no: 02

Given, MIPS instruction

lw $4, X($5)

And the base address of that array is 256 in decimal.

Now The value of X is in 128-bit architecture system. So, the register file will be of ~~182~~ 128 bits.

So in that case, storing each 128 bit will require, $\frac{128}{8}$ = 16 slots

So, A[5] is given,

$\therefore$ A[5] = slots × 5 × ($5)

= 16 × 5 ($5)

= 80 ($5)

$\therefore$ X = 80 A

Ans: to the que no: 03

Given,

x → $s0	we know
Y → $s1	$t0 – $t7 → $8 – $15
z → $s2	$s0 – $s7 → $16 – $23

In the solutions, I will write the full MIPS code first then I will mention the instruction type and line by line machine code.

For $X = 3Y + 5Z + 10$

MIPS code:

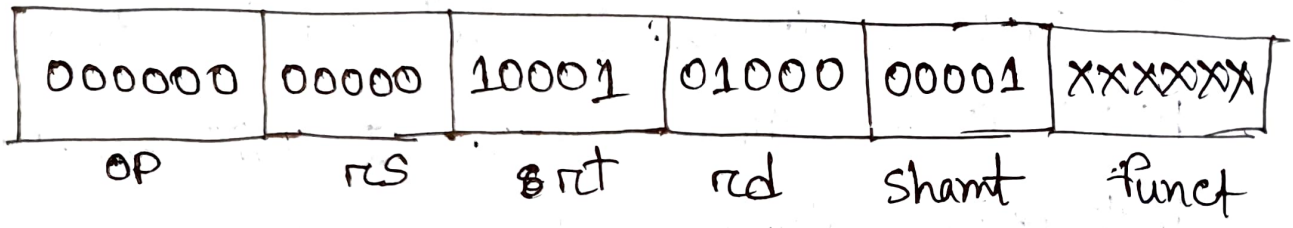
```
sll $t0, $s1, 1      # 2Y → $t0
add $t0, $t0, $s1    # 3Y → $t0
sll $t1, $s2, 2      # 4Z → $t1
add $t1, $t1, $s2    # 5Z → $t1
add $t0, $t0, $t1    # 3Y + 5Z → $t0
addi $s0, $t0, 10    # 3Y+5Z+10 → $s0
```

Machine Code:

① sll is R Type Instruction.

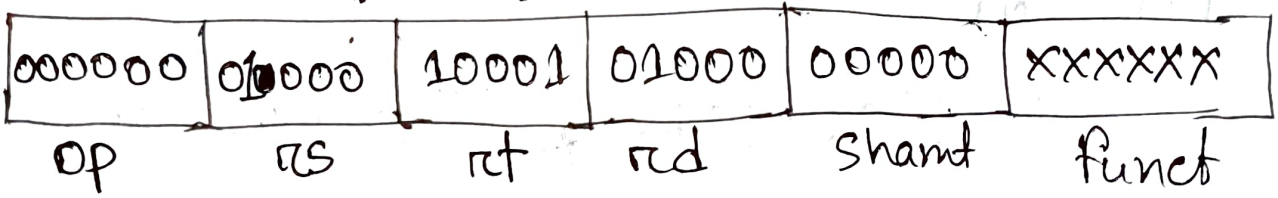
∴ sll $t0, $s1, 1

↳ sll $8, $17, 1



② add is R Type Instruction

add $8, $8, $17



③ sll $t1, $s2, 2

↳ sll $9, $18, 2

sll is R Type instruction

000000	00000	10010	01001	00010	XXXXXX
op	rs	rt	rd	shamt	funct

④ add $t1, $t1, $s2

↳ add $9, $9, $18

add is R Type Instruction

000000	01001	10010	01001	00000	XXXXXX
op	rs	rt	rd	shamt	funct

⑤ add $t0, $t0, $t1

↳ add $8, $8, $9

R Type

000000	01000	01001	01000	00000	XXXXXX
op	rs	rt	rd	shamt	funct

⑥ addi $s0, $t0, 10

↳ addi $16, $8, 10

addi is I-type instruction

XXXXXX	01000	10000	0000000000001010
op	rs	rt	Constant (16 bit)

Now For: $Y = 4X + Arr[10] + 2Z$

MIPS Code:

Arr → $s4

```
sll  $t0, $s0, 2      # 4X → $t0
sll  $t1, $s2, 1      # 2Z → $t1
add  $t0, $t0, $t1    # 4X+2Z → $t0
~~add~~
lw   $t1, 40($s4)     # 40($s4) → $t1
add  $s1, $t0, $t1    # 4X+Arr[10]+2Z → $s1
```

$s1 ↓ Y

Machine Code:

#1 sll $t0, $s0, 2

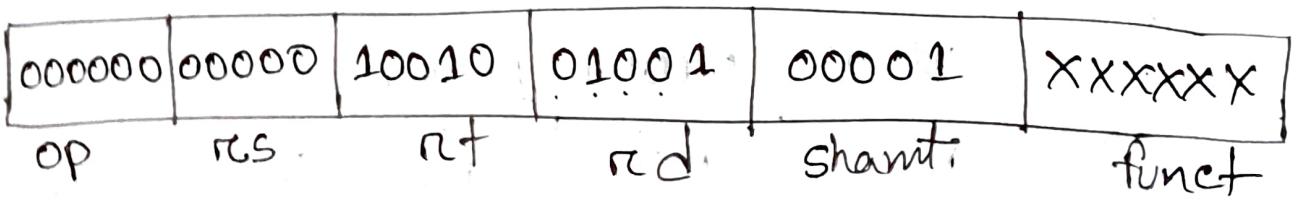
↳ sll $8, $16, 2

"R" Type

000000	00000	10000	01000	00010	XXXXXX
op	rs	rt	rd	shamt	funct

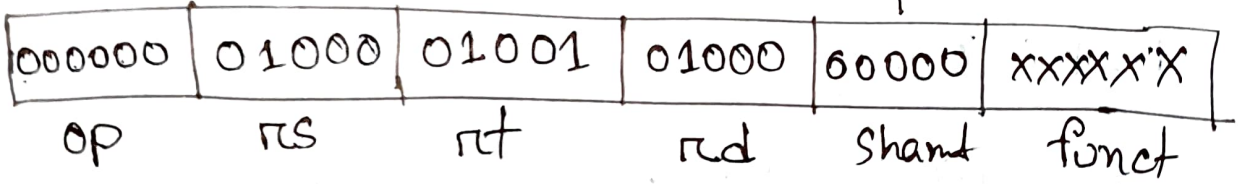
#2 sll $t1, $s2, 1 | R Type

↳ sll $9, $18, 1



#3 add $t0, $t0, $t1 | R Type Instructio

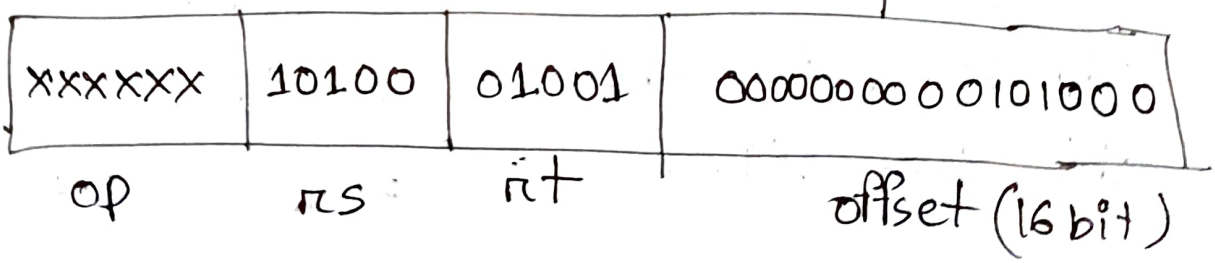
↳ add $8, $8, $9



#4 lw $t1, 40($s4)

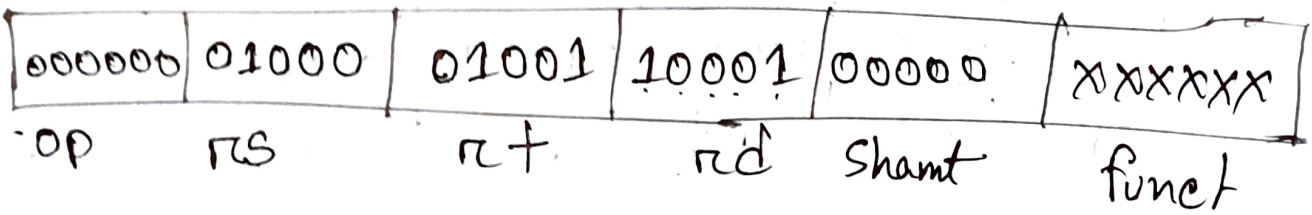
↳ lw $9, 40($20)

lw is I type instruction



#5 add $S1, $t0, $t1 | R Type

↳ add $17, $8, $9



Now for

$$Z = 5X + 5Y + Z$$

MIPS Code:

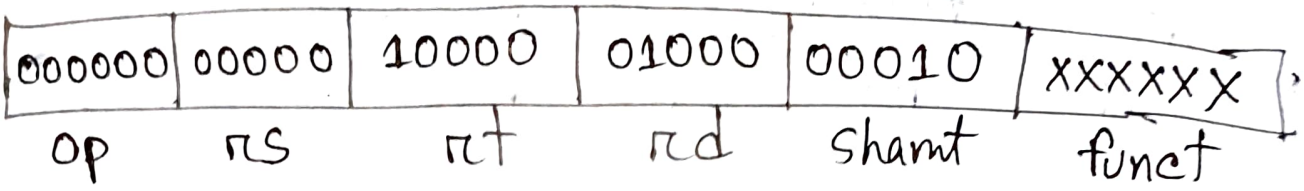
```
sll  $t0, $S0, 2        # $t0 → 4X
add  $t0, $t0, $S0      # $t0 → 5X
sll  $t1, $s1, 2        # $t1 → 4Y
add  $t1, $t1, $s1      # $t1 → 5Y
add  $t1, $t1, $t0      # $t1 → 5X + 5Y
add  $s2, $t1, $s2      # $s2/Z → 5X+5Y+Z
```

Machine Code:

#1 sll $t0, $s0, 2

↳ sll $8, $16, 2

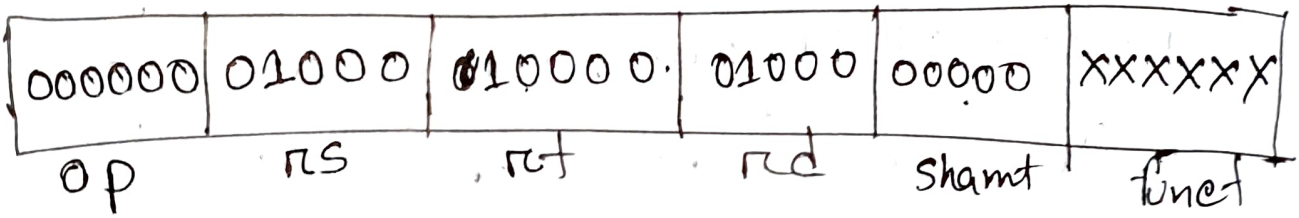
sll is R type



add $t0, $t0, $s0

↳ add $8, $8, $16

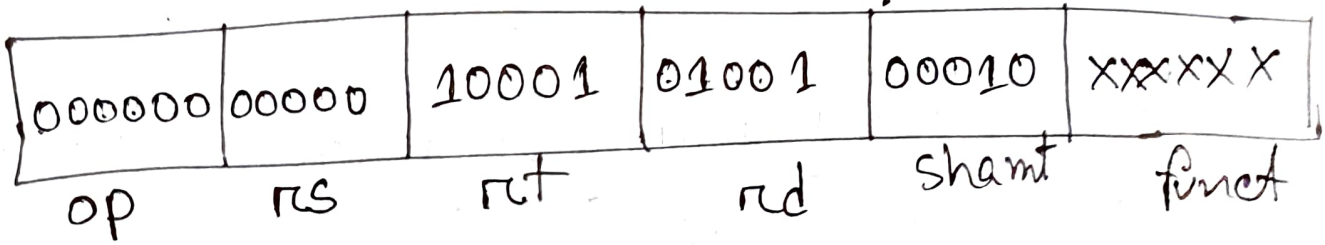
R type



sll $t1, $s1, 2

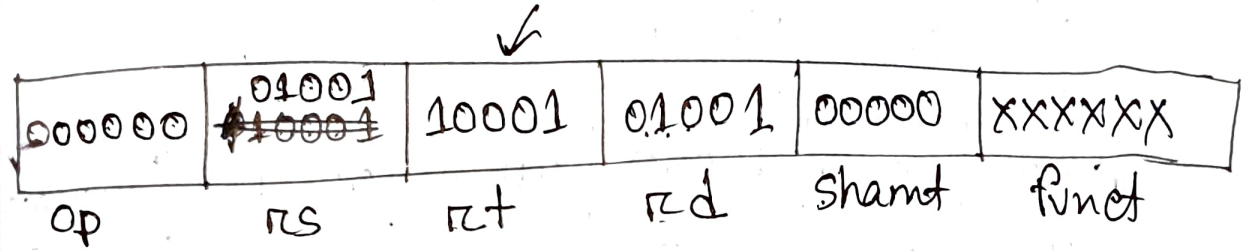
↳ sll $9, $17, 2

R Type



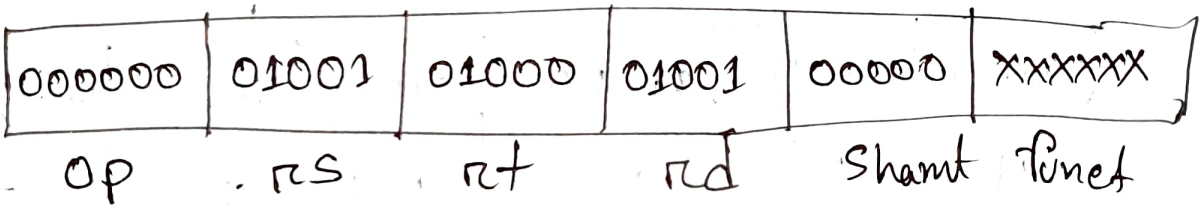
add $t1, $t1, $s1 | R type

↳ add $9, $9, $17



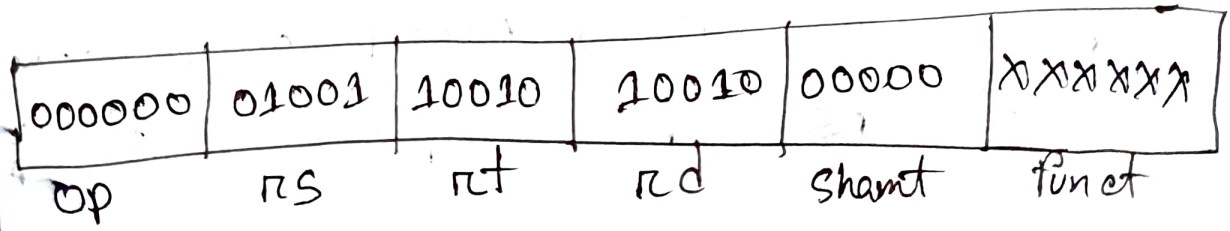
add $t1, $t1, $t0 | R type

↳ add $9, $9, $8



add $s2, $t1, $s2 | R type

↳ add $18, $9, $18



Now for $Arr[10] = X + Y - 30$

MIPS code:

```
add  $t0, $S0, $S1      # $t0 → x + Y
addi $t0, $t0, -30      # $t0 → x+Y-30
lw   $t1, 40($S4)       # $t1 → Arr[10]
sw   $t0, 40($S4)       # x+Y-30 → Arr[10]
```

Machine code:

```
# add  $t0, $S0, $S1
↳ add  $8, $16, $17
```

R Type

000000	10000	10001	01000	00000	xxxxxx
op	rs	rt	rd	shamt	funct

```
# addi $t0, $t0, -30
↳ addi, $8, $8, -30
```

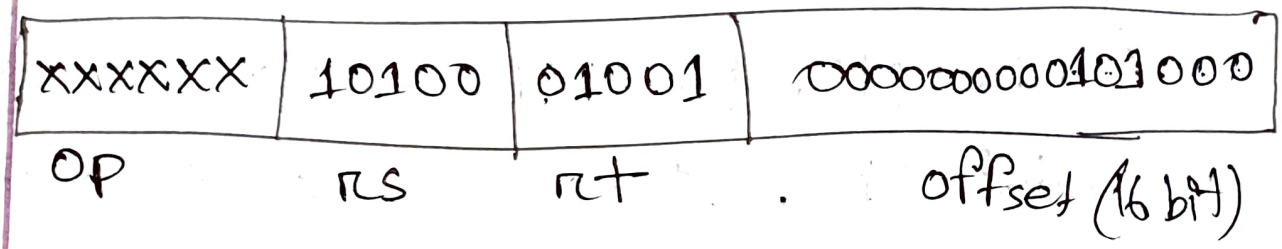
I Type Instruction

xxxxxx	01000	01000	1111111111110110
op	rs	rt	Const(-30)

2's

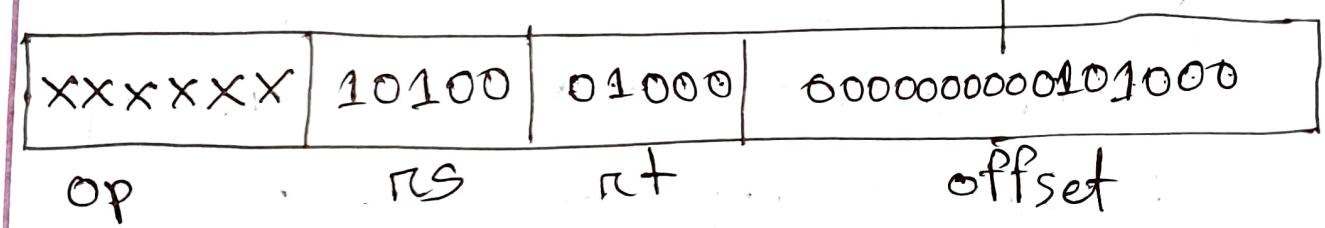
lw $t1, 40($s4) | I type

↳ lw $9, 40($20)



sw $t0, ~~40($s0)~~ 40($s4) | I type

↳ sw $9, 40($20)



Ans: to the que no: 04

Given,

Base of B is in $s0

i → $s1

f → $s2

Given,

f = B[i]

MIPS code:

```
sll $t0, $s1, 2
add $t1, $t0, $s0
~~lw $s2~~
lw $s2, 0($t1)
```

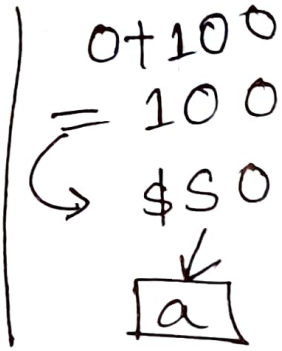
Ans: to the que no: 05

As MIPS does not have any assignment instructions, we have to find a way out. Now, we know we can add. So, if we add the integer and $zero to that we could do the work that we intended.

Now the MIPS code:

Let a be in $S0

```
addi $S0, $zero, 100
```



So, even though we did not use any assign instruction, we have assigned the value.

Ans: to the que no: 06

Firstly, We know, MIPS Follows Big Endian.

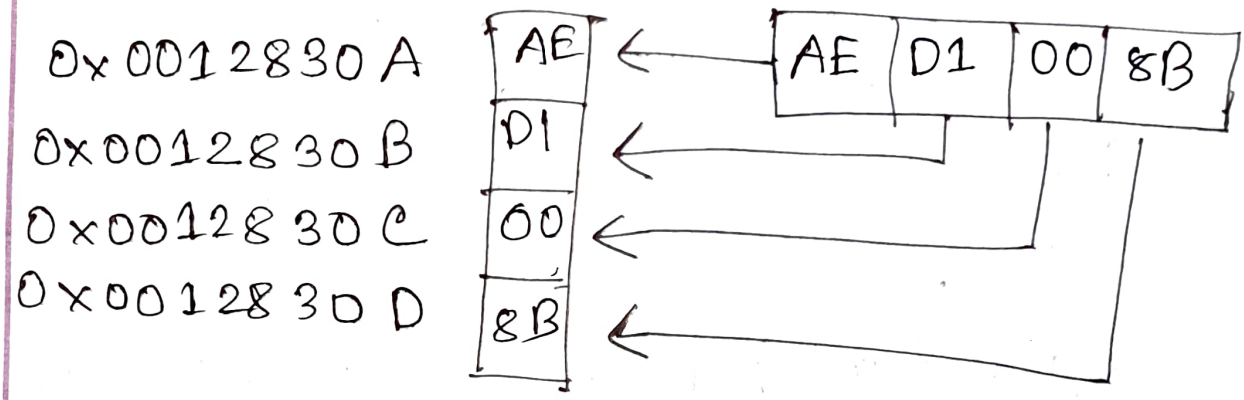
Big Endian: MSB is in least or lowest address position. LSB in Highest address.

Little Endian: LSB is in ~~los~~ lowest and MSB is in highest memory address.

Given data : AED1008B

Given memory address : 0x00128301A to 0x00128301D

So, for Big Endian MSB will be in lowest following the below diagram



For Little Endian: