

Assignment 1

Section: 09

① CSE340

G.M. Anjot Rahman

19101498

1. In a code (in C/Java/Python etc) we have multiple lines each containing some operations. When these codes get converted into MIPS Instruction (can be R or I J type), the CPU needs to keep track which line of instruction is getting executed and what will be executed next. For this task CPU has a register called program counter. It holds the address of current instruction. On the other side, MIPS register \$0 or \$zero is the constant 0. 40 registers commonly hold 0, it can not be overwritten. Beside this, we can use \$0 register in various instructions. For instance, for moving values between register and \$0, \$50, \$zero. But program counters can not be used in any type of instructions.

(2)

Main memory contains portions on blocks of 8 bits.

So, for sequential instruction execution;

$$\frac{64}{8} = 8 \text{, and } \frac{128}{8} = 16 \text{ will be the}$$

increment, starting on instruction 0, and offset

Q. We know, in 8-bit, increment \times offset (base address)

destination

increment for 128-bit architecture, $128/8 = 16$

offset in $A[5]$ $\Rightarrow 5 \text{ no. of bits}$

then, $x = \text{increment} \times \text{offset}$

$$= 16 \times 5 = 80 \text{ (Ans)}$$

(3)

 $x \rightarrow \$50$ $y \rightarrow \$51$ $z \rightarrow \$52$

$$x = 3y + 5z + 10$$

2.

 $3y \Rightarrow \text{all sto, } \$51, 1 \text{ -R type}$

000 000	000 000	551, 513 10001	450, 48 01000	00001	$x \text{ } xx \text{ } xxx$
op code	10	14	10	short	func

~~551, 513~~

 $add \text{ sto, sto, } \$51 \text{ -R type (3y)}$

000 000	sto, 48 01000	551 10001	450, 48 01000	000 000	$xxx \text{ } xxx$
op code	10	14	10	short	func

 $from \ 52 \Rightarrow \text{all } st1, st1, \$52, 2 \text{ -R type}$

000 000	000 000	552, 518 100010	451, 49 01001	000 10	$xxx \text{ } xxx$
op code	10	14	10	short	func

 $add \ st1, st1, \$52 \text{ -R type (5z)}$

000 000	st1 01001	552 100010	451 01001	000 000	$xxx \text{ } xxx$
op code	10	14	10	short	func

 $add \ sto, sto, st1 \text{ -R type (3y+5z)}$

000 000	sto 01000	st1 01001	sto 01000	000 000	$xxx \text{ } xxx$
op code	10	14	10	short	func

(4)

add \$10, \$10, 10 - I-type

xxxxxx	\$10 01000	\$10 \$50 10000 01000	0000 0000 0000 1010
opcode	rs	rt	rd offset

Jor: $r = 4x + \text{Addr}[10] + 28$

Jm 28 \Rightarrow \$11 \$10, \$52 - I-type

000 000	000 00	\$52 10010	\$10 01000	00001	func xxxxxx
opcode	rs	rt	rd	shamt	

Jm $4x \Rightarrow$ \$11 \$10, \$50, 2 - R-type

000 000	000 00	\$50 10000	\$11 01001	00010	xxxxxx
opcode	rs	rt	rd	shamt	func

Jm $\text{Addr}[10] \Rightarrow$

1W \$12, \$x10 (009) - I-type

xxxxxx	\$12 10100	\$12 01010	0000 0000 0010 1000
opcode	rs	rt	off

$4x + ARR[10] \Rightarrow add, g+1, g+2, 4+2$ - R-type

000 000	g+1 01001	g+2 01010	g+1 01001	000 000	XXX XXX
opcode	rd	rt	rd	shamt	junct

Now, $y = 4x + ARR[10] + 2z \Rightarrow$

$add, \del{g+1}, g+1, 4+0$ - R-type

000 000	g+1 01001	g+0 01000	g+0 01001	000 000	XXX XXX
opcode	rd	rt	rd	shamt	junct

Now, $z = 5x + 5y + 2$

$5x \Rightarrow 511, 4+0, \$80, 2$ - R-type

000 000	000 000	480 10000	4+0 01000	000 10	XXX XXX
opcode	rd	rt	rd	shamt	junct

$add, 4+0, 5+0, \$80$ - R-type

000 000	4+0 01000	5+0 10000	4+0 01000	000 000	XXX XXX
opcode	rd	rt	rd	shamt	junct

(6)

$5x = 51$ $\$71, \1 , 2 R-type

000 00	000 00	$\$81/10001$	$\$71$ 01001	000 10	xxxxxx
opcode	111	11	11	10	shamt funct

add $\$71$ $\$71, \01 R-type

000 00	$\$71$ 01001	$\$81$ 10001	$\$71$ 01001	000 00	xxxxxx
opcode	111	11	11	10	shamt funct

$5x+5y = 9d$ $\$70, \$70, \$71$ R-type

000 00	$\$70$ 01000	$\$71$ 01001	$\$70$ 01000	000 00	xxxxxx
opcode	111	11	11	10	shamt funct

$$\text{Now, } Z = 5x + 5y + Z$$

add $\$52, \$70, \$52$ R-type

000 00	$\$70/01000$	$\$52$	$\$52$	000 00	xxxxxx
opcode	111	11	11	10	shamt funct

7

$$\text{Now, } \text{Addr}[30] = X+Y-30;$$

$X+Y \Rightarrow \text{odd}$ \$to, \$so, \$s1 R-type

000 000	000 10000	001 10001	010 01000	000 000	XXX XXX
Opcode	rs	rt	rd	shamt	Junct

$X+Y-30 \Rightarrow$ odd \$to, \$to, -30 I-type

XXX XXX	010 01000	010 01000	1111 1111 1110 0010
Opnd	rs	rt	offset/immediate

binary of 30 in 16 bits

0000 0000 0001 1110

30 : 1111 1111 1110 0001
complement

$+1$

$\overline{1111 1111 1110 0010}$ (-30 using 2's complement)

$$\text{Now, } \text{Addr}[30] = X+Y-30 \Rightarrow$$

sw \$to, 4x10(\$s1) \rightarrow I-type

Q3

xxx xxx	\$54 10100	\$70 01000	40 0010 1000
opcode	RS	RT	offset

4. Base of B[i] \rightarrow \$50, $i \rightarrow$ \$52, $f \rightarrow$ \$52

$$f = B[i]$$

add \$71, \$50, \$51

~~sub~~ \$52, \$52, 0(\$71)

add \$52, \$zero, \$72

⑨

5. MIPS does not have an assignment instruction.

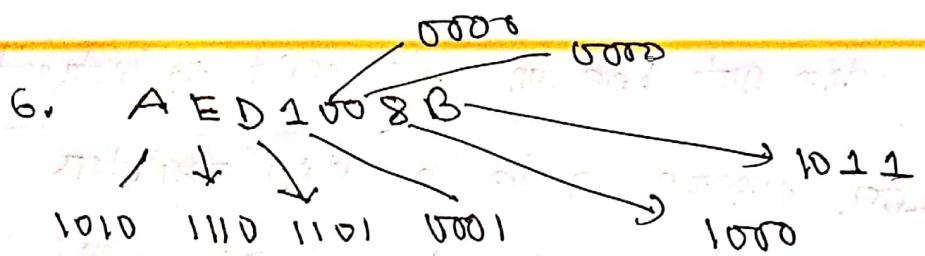
int a = 100. Suppose a is in \$00 register. We have a \$zero register. We know addi instruction can add a value in a register and from a constant integer. Here constant is 100. So

$$0 + 100 = 100. \text{ So}$$

addi \$00, \$zero, 100
+ distinction source 1 source 2 (constant)
keyword

Here addi will add value in \$zero with 100 and will store in register \$00 which is the register of variable a.

10



We know, every 4 bit of hexadecimal is equivalent to 4 bit binary. As we know main memory is ~~so it~~ if divided into 8 bit slot, then 32 bit data will be stored in 4 different slots in main memory. $AE = 8 \text{ bit}$, $D1 = 8 \text{ bit}$, $00 = 8 \text{ bit}$ and $8B = 8 \text{ bit}$.

(i) Big Endian: In Big Endian, most-significant byte get stored in least address of a word. Here $0x0012830A$ is the least address of the given memory address, so in:

$0x0012830A \rightarrow AE$ or MSB byte will be stored

$0x0012830B \rightarrow D1$, $0x0012830C \rightarrow 00$ and

LSB 8B will get stored in $0x0012830D$.

11

(ii) Little Endian:

In Little Endian order, LSB byte get stored at
last address of a word.

88 88 → 0x0012830A

00 00 → 0x0012830B

1D 01 → 0x0012830C

EA 0A → 0x0012830D