

# Computer Architecture

Fall, 2019

Week 5

2019.10.7

## [group9]

1. [Multiple Choice]. Choose the correct statement.
  - (A) sb,sh are the instructions dealing with “byte”/“half word”. Moreover, they store leftmost “byte”/“half word” data from register to RAM.
  - (B) Assume that we use sb instruction to store data “A7”; due to signed extension, the data will be stored as “AAAAAAA7”.
  - (C) Assume that we use lbu instruction to load data “A7”, the data will be stored as “FFFFFFA7”.
  - (D) Assume that we use sb instruction to store data “A7”; due to signed extension, the data will be stored as “FFFFFFA7”.

**Ans: (D)**

- (A) leftmost -> rightmost**
- (B) “AAAAAAA7” -> “FFFFFFA7”**
- (C) “FFFFFFA7” -> “000000A7”**
- (D) is correct**

## [group12] (對抗賽)

2. 在下列五種 MIPS Addressing Mode 中，請簡述各個 mode 取得 data 的方式。
  1. Immediate addressing
  2. Register addressing
  3. Base addressing
  4. PC-relative addressing
  5. Pseudodirect addressing

**Ans:**

- 1. 直接在 immediate 裡面**
- 2. 透過 rs rt rd 取得 reg 編號 再去對應的 reg 裡面拿**
- 3. 透過描述的編號去對應的 reg 裡面拿到 base address 後 再加上一個 offset(immediate 的值)去到 memory 拿資料**
- 4. Program counter 加上 immediate 的值得到 memory 位址再取得資料**
- 5. Program counter 的前 4 個 bit 連接 instruction 的地址得到 memory 位址後，再去到 memory 這個位址拿**

## [group1] (對抗賽)

3. In the execution of a procedure, the program must follow some steps. What is the correct order of these 6 steps?

- A. Acquire the storage resources needed for the procedure.
- B. Perform the desired task.
- C. Put the result value in a place where the calling program can access it.
- D. Put parameters in a place where the procedure can access them.
- E. Transfer control to the procedure.
- F. Return control to the point of origin, since a procedure can be called from several points in a program.

**Ans:**

**D -> E -> A -> B -> C -> F**

**[group11] (對抗賽)**

4. If the current value of the PC is 0x00000600, can you use a single branch instruction to get to the PC address=0010 0000 0000 0001 0100 1001 0010 0100<sub>2</sub>?

**Ans:**

**No. Branch is i-format.**

opcode(6 bits)	rs(5 bits)	rd(5 bits)	offset(16 bits)
----------------	------------	------------	-----------------

**so, the maximum distance to branch is  $\pm 2^{15}$**

**Rewrite 0010 0000 0000 0001 0100 1001 0010 0100<sub>2</sub> to hexadecimal**

**0010 0000 0000 0001 0100 1001 0010 0100<sub>2</sub>=0x2001492416**

**The difference between current address and destination address**

**= 0x2001492416-0x0000060016=0x2001432416=53695363610 $>2^{15}$ =32768**

**Hence, we can't use a single branch instruction to get to the PC address=0010 0000 0000 0001 0100 1001 0010 0100<sub>2</sub>.**

**[group10]**

5. 當 caller 呼叫 subroutine 時 \$ra 暫存器會存放 caller program 的返回位置，請問若 caller 呼叫 subroutine\_1，此時\$ra 存放 caller 返回位置，subroutine\_1 呼叫 subroutine\_2，此時\$ra 存放 subroutine\_1 返回位置，subroutine\_2 呼叫 subroutine\_3，此時\$ra 存放 subroutine\_2 返回位置，請問(1)subroutine 全部執行完畢，要用什麼機制回到一開始 caller 呼叫 subroutine\_1 的返回記憶體位置？(2)此機制有什麼特性？

**Ans:**

**(1) caller 必須將\$ra 暫存器 push 到 stack 中，當完成 subroutine 後 pop 出\$ra。**

**(2) Stack 設計為後進先出，剛好符合比較晚被呼叫到的 subroutine 先執行完**

**[group6] (對抗賽)**

6. Fill in the blanks below to practice with target addressing.

addi \$s4, \$zero, 1	80000	8	0	17	1
----------------------	-------	---	---	----	---

bne \$s1, \$s4, C2_COND	80004	5	17	20	(a)
j C1_BODY	80008	2	(b)		
C2_COND: addi \$s4, \$zero, 2	80012	8	0	17	2
bne \$s1, \$s4, C3_COND	80016	5	17	20	(c)
j C2_BODY	80020	2	(d)		
C3_COND: addi \$s4, \$zero, 3	80024	8	0	17	3
bne \$s1, \$s4, EXIT	80028	5	17	20	(e)
j C3_BODY	80032	2	(f)		
C1_BODY: addi \$s1, \$s1, 1	80036	8	17	17	1
j EXIT	80040	2	(g)		
C2_BODY: addi \$s1, \$s1, 2	80044	8	17	17	2
j EXIT	80048	2	(h)		
C3_BODY: addi \$s1, \$s1, 3	80052	8	17	17	3
EXIT: ...	80056				

**Ans:**

**(a) 1**

**(b) 20009**

**(c) 1**

**(d) 20011**

**(e) 6**

**(f) 20013**

**(g) 20014**

**(h) 20014**

[group8] (對抗賽)

7.

beq \$s0,\$s1,L1

...

...

...

L1:

假設在以上 MIPS 指令中的 L1，其相對於 beq 指令的記憶體位址超過 16 位元所可以表示的距離，則我們可以如何改寫上述指令來達成原本的目的？

**Ans:**

bne \$s0,\$s1,L2

j L1

L2:...

...

...

L1:

[group3] (對抗賽)

8. 下列敘述哪些正確，若有錯請改出錯誤

- (A) Leaf procedure doesn't call any other procedure
- (B) Use `lb $rt, offset($rs)`, 11 1111 0001 is sign extended to 32 bit in \$rt, the new representation is 0000 0011 0001
- (C) This is a jump address. The first 4 bits of the program counter are 1111. If target field = 00 0000 1100 0011 1110 0101 1011, the 32-bit address = 1100 0000 0011 0000 1111 1001 0110 1100.
- (D) Design principles: (1)Simplicity favors regularity. (2)Smaller is faster. (3)Make the common case fast. (4)Good design demands good compromises.

Ans: (A) (D)

數  
最  
前  
回  
應  
該  
為  
而  
不  
是

[group4]

```
void strcpy (char x[], char y[]){
```

```
int i;
```

```
i = 0;
```

```
while ((x[i] = y[i]) != '\0')
```

```
i += 1; }
```

What is the MIPS assembly code? Please fill in the blanks.

```
addi $sp, ____, ____
```

```
sw $s0, 0($sp)
```

```
add $s0, $zero, ____
```

```
L1:
```

```
add $t1, ____, ____
```

```
lbu $t2, 0($t1)
```

```
____ $t3, $s0, $a0
```

```
sb $t2, __($t3)
```

```
beq $t2, ____, L2
```

```
addi $s0, $s0, 1
```

```
j ____
```

```
L2:
```

```
lw $s0, 0($sp)
```

```
addi $____, $____, 4
```

\_\_\_ \$ \_\_\_

**Ans:**

**addi \$sp, \$sp, -4**

**sw \$s0, 0(\$sp)**

**add \$s0, \$zero, \$zero**

**L1:**

**add \$t1, \$s0, \$a1**

**lbu \$t2, 0(\$t1)**

**add \$t3, \$s0, \$a0**

**sb \$t2, 0(\$t3)**

**beq \$t2, \$zero, L2**

**addi \$s0, \$s0, 1**

**j L1**

**L2:**

**lw \$s0, 0(\$sp)**

**addi \$sp, \$sp, 4**

**jr \$ra**