Computer Architecture

**Fall, 2024**
**Week 5**
**2024.09.30**

**組別：_____ 簽名：_____**

[group 3]

1. Qustion:

a. What is the range of addresses for conditional branches in MIPS?

b. What is the range of addresses for jump and jump and link in MIPS?

Ans:

a. ±128KB (2^15-1 ~ -2^15 words)

b. Anywhere within a block of 256MB addresses where PC supplies the upper 4 bits

[group 4]

2. Question:

The following problem deal with translating from C to MIPS. Assume that the variables x, y,z ,and result are assigned to registers $s0, $s1, $s2, and $s3, respectively. What is the corresponding MIPS assembly code?

```
int f(a,b,c){
    return (2*a)-(b+c);
}
result= f(x,y,z);
```

Ans:

```
add $a0, $zero, $s0
add $a1, $zero, $s1
add $a2, $zero, $s2
jal f
add $s3, $zero, $v0
f:
addi $sp, $sp, -8
sw $t0, 0($sp)
sw $t1, 4($sp)
sll $t0, $a0, 1
```

```
add $t1, $a1, $a2
sub $v0, $t0, $t1
lw $t0, 0($sp)
lw $t1, 4($sp)
addi $sp, $sp, 8
jr $ra
```

[group 5]

3. Question:

Convert the C code below into MIPS

C code:

```
int fibonacci(int n) {
    if (n <= 1) {
        return n;
    }
    else {
        return fibonacci(n - 1) + fibonacci(n - 2);
    }
}
```

Ans:

```
fibonacci:

    addi $sp, $sp, -8

    sw $ra, 0($sp)

    sw $a0, 4($sp)

    slti $t0, $a0, 2

    beq $t0, $zero, L1

    add $v0, $v0, $a0

    addi $sp, $sp, 8

    jr $ra

L1:

    addi $a0, $a0, -1
```

```
jal fibonacci

addi $a0, $a0, -1

jal fibonacci

lw $a0, 4($sp)

lw $ra, 0($sp)

addi, $sp, $sp, 8

jr $ra
```

[group 7]

4. Question:

If the current value of the program counter (PC) is 0x20000603, can a single branch instruction reach the PC address 0001 1111 1111 1111 1111 1001 0010 $0100_2$?

Ans:

**Yes.** Branch is i-format, so the maximum distance to branch is $+-2^{15}$
Rewrite 0001 1111 1111 1111 1111 1001 0010 $0100_2$ to hexadecimal
0001 1111 1111 1111 1111 1001 0010 $0100_2$=0x1FFFF924
The difference between current address and destination address = 0x20000603 - 0x1FFFF924=0x00000CDF<$2^{15}$

[group 8]

5. Question:

In the execution of a procedure, the program must follow some steps. What does the correct order of the 6 steps happen in?

A. Acquire the storage resources for the procedure.
B. Perform procedure's operations.
C. Place the result value in a register that the calling program can access.
D. Place parameters in a register where the procedure can access them.
E. Transfer control to the procedure.
F. Return to place of call.

Ans:

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

[group 10]

6. Question:

(1.)請問以下 immediate 的單位分別為何?

ld $t0,1200($t1)
lw $t0,1200($t1)
lh $t0,1200($t1)
lb $t0,1200($t1)

(2.)

0x00008000 bne $t0,$zero,Label1
...
...
0x00008024 Label1

請問在 0x00008000 的 bne 指令中，immediate 應該為多少(用 binary 表示)?單位是 byte 還是 word?

(3.)承第 2 題，已知會跳的情況下，若 bne 改為 jump，則 target address 的位置應該填多少(用 binary 表示)?單位是 byte 還是 word?

Ans:

(1.) 這裡的 immediate 是以 byte 為單位，不隨讀取資料長度而有所影響

(2.) 2,3 題的 immediate 都是以 word 為單位

0x00008000 與 0x00008024 相差 36 個 byte，又因 PC 已經先加 4byte，

故 immediate = (36-4)/4 = 8 個 word = 0b 0000 0000 0000 0100

(3.) 0x00008024 = 0b 0000 0000 0000 0000 1000 0000 0010 0100

而 target address = 0000 || immediate || 00

所以 immediate = 0b 0000 0000 0000 1000 0000 0010 01

[group 11]

7. Question:

In the MIPS architecture, when executing a non-leaf procedure call, why is it necessary to save parameters and registers in addition to saving the return address $ra? What problems would arise if they were not saved?

Ans:

In a non-leaf procedure call in MIPS, it's necessary to save parameters and registers because the procedure may call other functions that change their values. Without saving them, the outer procedure would lose its original data, causing errors. Saving these values on the stack ensures they can be restored correctly after the nested calls.

[group 12]

8. Question:

Given the following characteristics, identify which belongs to a leaf procedure and which to a non-leaf procedure:

1. **Characteristic A**: The procedure does not call any other procedures.
2. **Characteristic B**: The procedure may call other procedures.
3. **Characteristic C**: Typically has simpler stack management because it does not need to save space for additional return addresses.
4. **Characteristic D**: When calling other procedures, it needs to save the return address and any arguments or temporaries needed after the call on the stack, and restore them from the stack after the call.

Ans:

A，C　leaf procedure

B，D non-leaf procedure

[group 13]

9. Question:

What are the condition codes used in ARM architecture for the result of an arithmetic or logical instruction, and how are they utilized in compare instructions?

Ans:

The condition codes used in ARM architecture are Negative, Zero, Carry, and Overflow. These codes are set based on the result of an arithmetic or logical instruction. Moreover, their top 4 bits of instruction word in each instruction are condition value, which can avoid branches over single instructions.

[group 14]

10. Question:

Choose the correct answer(s) for the following statements:

(A) Caller is in charge of placing parameter in registers, transferring control to procedure, which acquires
storage for procedure
(B) There are 32 registers can be defined freely by the programmer
(C) We have one register file in caller and callee, so when using it, what we only need to do is storing the
former data in a stack in memory
(D) We can implement a recursive function in non-leaf procedure

Ans:

D
(A) Caller doesn’t acquire the storage for procedure
(B) Some registers are defined in hardware and programmer cannot redefine them such as $zero
(C) also need to reload it back to the restore
(D) is correct, see example in p.89 on the lecture slide

[group 9]

11. Question:

```
lb $t1, 0($t0)
lbu $t2, 0($t0)
```

The data in 0($t0) is A8. Write down the contents of $t1 and $t2.

| |
|---|
| $t1: |
| $t2: |

Ans:

| | |
|---|---|
| $t1: AAAAAAA8 | 應為 FFFFFFA8 |
| $t2: 000000A8 | |