Computer Architecture

Fall, 2020

Week 6

2020.10.19

組別：_____　簽名：_____

[group13]

1. The following is MIPS code.

```
Loop: sub    $8 , $9 , $10
      beq    $9 , $0 , End -----→T
      add    $8 , $8 , $10
      addi   $9 , $9 , -1
      j      Loop
End:
   …
```

The address of instruction “beq $9 , $0 , End” is T(word address).

What is the byte address of the instruction “End”?

What is the PC-relative address in binary representation of the instruction “beq $9 , $0 , End” ?

Ans:

4T + 4 + (3*4)

decimal: 3 -> binary: 0000 0000 0000 0011

[group11] (對抗賽)

2. Consider the following code:

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

$t1 contains the address 0x10000000

Note the MIPS architecture utilizes big-endian addressing.

Assume that the data (in hexadecimal) at address 0x10000000 is 0xF12E3AB4.

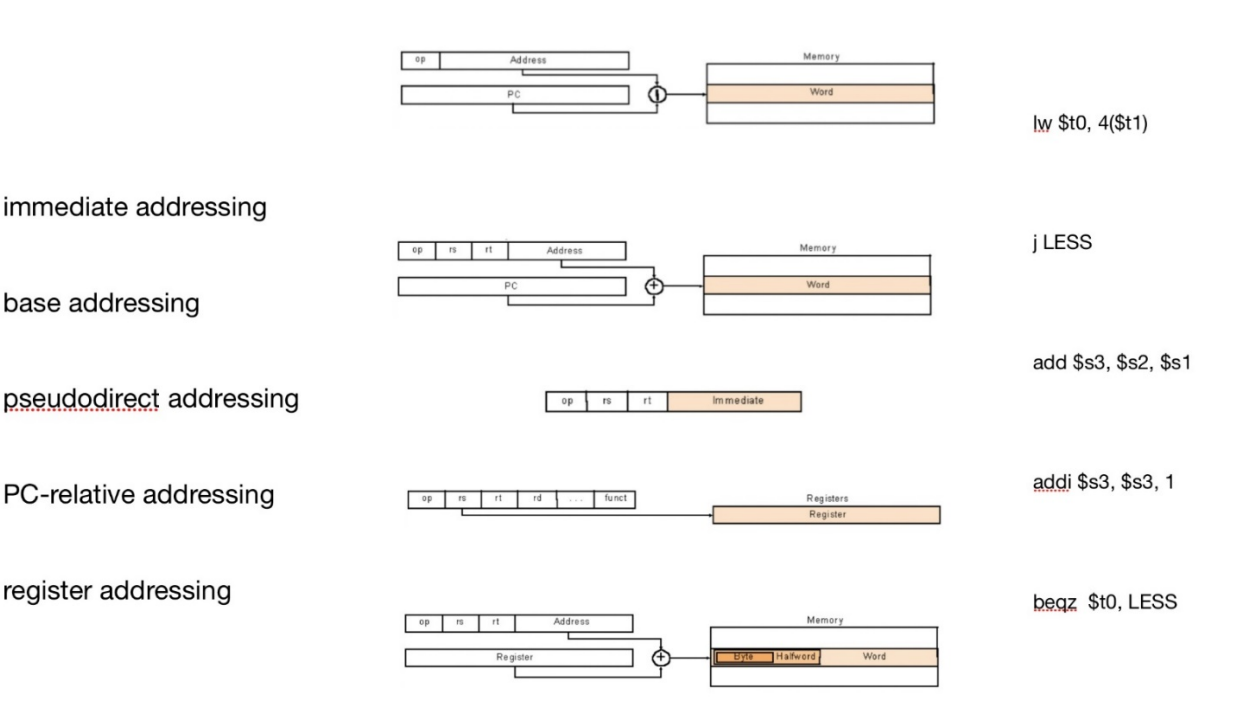
What is the value stored in register $t0?

Ans:

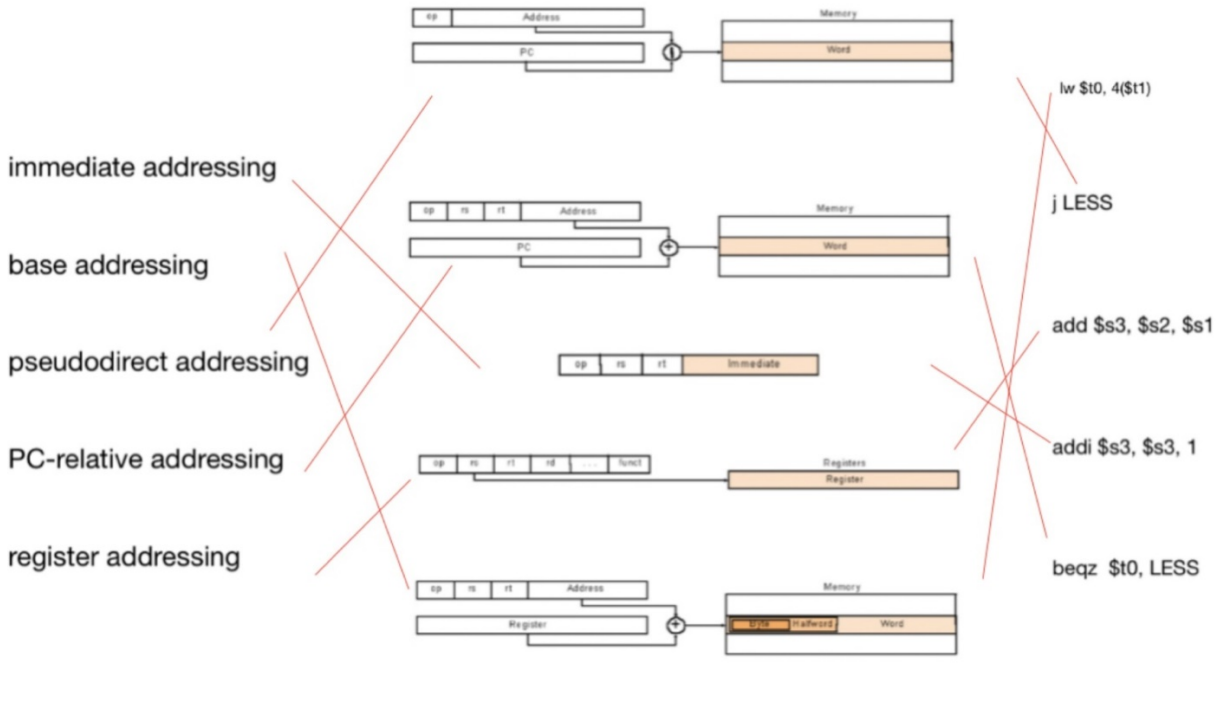
The Value in $t0 is 0xFFFFFFF1

[group5]

3. Map the MIPS addressing modes with correct images and examples



Ans:



[group10] (對抗賽)

4. Consider the following MIPS loop:

```
LOOP:   slt $t2, $0, $t1
        beq $t2, $0, DONE
        addi $s2, $s2, 2
        addi $t1, $t1, -1
        j LOOP
DONE:
        …
```

Assume that the registers $s1, $s2, $t1, and $t2 are integers A, B, i, and temp, respectively.
For the loops written in MIPS assembly above, assume that the register $t1 is initialized to the value N (N>0). How many MIPS instructions are executed?

Ans:

The code in C should be like:

```
        while(i != 0){
                B = B+2;
                i--;
}
```

Each full run of the loop contains 5 instructions, and the loop keeps running until i becomes 0. (It takes N times.)
After i becomes zero, only the first two instructions are executed and then it will goto DONE. So there are totally 5N+2 instructions executed.

[group6] (對抗賽)

5. True or False?

(a) For register addressing, the operands are constant.
(b) For immediate addressing, the operands are register.
(c) for PC-relative addressing, the address of operand is the sum of PC and sign-extended immediate.
(d) For Pseudo-direct addressing, the address of operand is 6 MSB of the current address followed by 26 bits in the instruction.

Ans:

(a) f, are register.
(b) f, there is a constant.
(c) f, the immediate will shift left for 2 bits also.
(d) f, the 4 MSB and 26 bits in the instruction, last 2 bits are 0.

[group3] (對抗賽)

6. 下列敘述何者錯誤，有錯的話請寫出錯在哪裡
   (a) A program counter is a register in a computer processor that contains the address (location) of the instruction being executed at the current time.
   (b) In bne and beq instruction, if branch taken, PC(program counter) wound jump to (PC + immediate*4)
   (c) PC-relative addressing mode specifies byte address
   (d) Because there is 6 bits in opcode ,so in j instruction of MIPS, we take the 6 highest order bits from the PC combine with the last 26 bits of j instruction to form the 32-bit address that we want to jump to.
   (e) Assume that we use sb instruction to store data “A7”; due to signed extension, the data will be stored as “AAAAAAA7”

Ans:

   (a) 錯，pc 存取的 instruction 是下一個將要執行的 instruction。不是現在正在執行的 instruction。
   (b) 要改成(PC+4+immediate)，因為執行完此指令以後，PC 會先往下一個指令跳
   (c) specifies byte address → specifies word address
   (d) 6 highest order bits → 4 highest order bits
   (e) Stored data is F7. No sign extension. (Store instruction doesn’t need to perform sign extension)

[group14] (對抗賽)

7. In MIPS, can we use conditional branches or unconditional branch jump to below address of instruction? (PC= 0X20000000)

a. 0X00001000
b. 0x20001400

Please give an explanation.

Ans:

conditional branches: branch (Range: -2^17+4~2^17)
unconditional branch: jump (Range: PC and destination address have to in the same block that means high-order 4 bits have to be same.)

a. Because 0X00001000-0X20000000=0XE0001000
   0XE0001000<-2^17+4=0XFFFF0004(branch can’t.)
   Jump can’t.

b. Because 0X20001400-0X20000000=0X00001400
   0X00001400<2^17=0X00008000-1(branch can.)

Jump can.

[group4] (對抗賽)

8. Please turn below MIPS code into C code.

```
main:
    addi   $a0, $0, 1
    addi   $a1, $0, 2
    addi   $a2, $0, 3
    addi   $a3, $0, 4
     jal     EXAMPLE
    add   $s4, $0, $v0
EXAMPLE:
      addi   $sp, $sp, -12
    sw     $s0, 0($sp)
    sw     $t0, 4($sp)
    sw     $t1, 8($sp)
    add    $t0, $a0, $a1
    add    $t1, $a2, $a3
    sub    $s0, $t0, $t1
    add    $v0, $s0, $zero
    lw     $s0, 0($sp)
    lw     $t0, 4($sp)
    lw     $t1, 8($sp)
    addi   $sp, $sp, 12
    jr     $ra
```

Ans:

```
    int EXAMPLE(a , b , c , d){
            int ans = (a + b) – (c + d);
             return ans;
    }
    void main(){
             int test = EXAMPLE(1 , 2 , 3 , 4);
    }
```

[group8] (對抗賽)

9. Please tell the following statements are true or false, if false, please also tell us why.
   a. the frame pointer is convenient because all references to variables in the stack within a procedure will have the same offset.
   b. Specify the immediate in words. Now, we can branch ± 2^17 words from the PC.
   c. In I-format, we can't use immediate to store the instruction address since it cannot specify entire address.
   d. When we do the non-leaf procedure, we store the argument and return address to the stack before we do the 'jal' instruction, then we should restore the argument and return address form the stack after we do the 'jr' instruction.
   e. PC-relative addressing can branch ±2^15 words from the PC.
   f. For a MIPS program, if the current value of the PC is 0x00000000, a single branch instruction can jump to the instruction in address 0x00131076 (tips:2^17=131072)

Ans:

(a)(e) T

(b) false, ±2^15

(c) false, we still can store the relative address into immediate

(d) false, we should restore the argument and return address form the stack BEFORE. we do the jr instruction.

(f) This yields a value in range [-2^17, 2^17 - 4] so [-131072, 131068]. But you must take into account that the value is added to the instruction after the branch so the actual range relative to the instruction itself is [-131072+4,131068+4].