Computer Architecture

Fall, 2024
Week 4
2024.09.23

組別：_____ 簽名：_____

[group 1]

1. Qustion:

Given register information:
$s0 = 0x00000000
$s1 = 0x00000064
$s2 = 0x12345678

Now translate the following MIPS instructions into C code:

```
        add     $t0, $zero, $s0
Loop:
        sll     $t1, $t0, 2
        add     $t1, $s2, $t1
        sw      $t0, 0($t1)
        addi    $t0, $t0, 1
        slt     $t2, $t0, $s1
        bne     $t2, $zero, Loop
Exit:
```

Ans:

We assign $s0 to variable i = 0, $s2 to base address of array A,and $s1 value =100 in decimal

C code is:

```
for(i = 0; i < 100; i++){
    A[i] = i;
}
```

[group 4]

2. Question:

In the following program code:

```
if (g < h)    goto Less;
```

To convert this into MIPS, we need to use two instructions like slt (set less than) and bne (branch if not equal), instead of a single blt (branch if less than). Why is MIPS designed this way?

Ans:

Hardware for <, >=, and similar operations is slower than for = and !=. Combining these operations with a branch requires more work per instruction, which would necessitate a slower clock. However, using a slower clock would penalize the performance of all instructions.

[group 5]

3. Question:

Assume that:

- The base address of the array is stored in register $s0.
- The variable sum is stored in register $s1.
- The variable i is stored in register $s2.

Given the following C code, write the corresponding MIPS assembly code:

```
while (array[i] != 0) {
   sum += array[i];
   i++;
}
```

Ans:

```
Loop:
    sll $t0, $s2, 2          # Calculate the address of array[i]
    add $t0, $t0, $s0
    lw   $t1, 0($t0)         # Load array[i] into $t1
    beq $t1, $zero, Exit # If array[i] == 0, exit the loop
    add $s1, $s1, $t1      # sum += array[i]
    addi $s2, $s2, 1       # i++
    j Loop
Exit:
```

[group 6]

4. Question:

Write down the answers and explain why:

a. Shift right arithmetic by 4 bits: 1001 0011 1110 1110 0101 1101 1001 1111

b. What is the relationship between Shift 和 multiplication? Why do we use shift instead of multiplication?

c. T/F : I-format is used for lw, sw, and register, such as add and sub.

d. T/F : The maximum and minimum immediate that can be implemented in I-Format are +- 2^16

e. T/F : Use beq and bne instead of blt, bge, etc is an example of good design compromise.

Ans:

a. 1111 1001 0011 1110 1110 0101 1101 1001

b. Multiply by 2^n equals to shift left by n. Shift -> O(n), multiple -> O(n^2), shift is faster.

c. F

I-format is used for immediate, lw, sw. R-format is used for register.

The I-format is used for instructions like lw (load word) and sw (store word) that involve memory addresses and immediates, but not for add and sub.

The add and sub instructions use R-format, which is specifically for operations that involve only registers and no immediate values.

d. F

The I-format uses a 16-bit immediate field. This means that the immediate range is actually -2^15 to 2^15 - 1.

e. T

[group 7]

5. Question:

Interpret the big-endian scheme and the little-endian scheme. If a 32-bit data, 0xd75afb68, is stored in memory from address 100, what is the memory content at address 101 for each scheme?

Address	Big endian	Little endian
100	d7	68
101		
102	fb	5a
103	68	d7

Ans:

Address	Big endian	Little endian
100	d7	68
101	5a	fb
102	fb	5a
103	68	d7

[group 8]

6. Question:

Consider the following MIPS code:
srl $t2, $t3, 3
1) What is the function of this instruction?
2) If the value in register $t3 is 0x000000F8 (hexadecimal), what will be the value in register $t2 after executing this instruction?

Ans:

1) The srl instruction stands for "Shift Right Logical." It shifts the value in register $t3 to the right by 3 positions, filling the leftmost bits with 0.
   (This is equivalent to dividing the value in $t3 by 8.)
2) The value in $t3 is 0x000000F8, which in binary is:
   0000 0000 0000 0000 0000 0000 1111 1000
   After shifting this value to the right by 3 bits, the result is:
   0000 0000 0000 0000 0000 0000 0001 1111
   Converting this binary value back to hexadecimal gives 0x0000001F.

Therefore, the value in register $t2 will be 0x0000001F.

[group 9]

7. Question:

Complete the following table:

Operation	C	Java	MIPS
Shift left	<<	<<	
Shift right	>>	>>>	
Bitwise AND	&	&	
Bitwise OR	|	|	
Bitwise NOT	~	~	

Ans:

Operation	C	Java	MIPS
Shift left	<<	<<	sll
Shift right	>>	>>>	srl
Bitwise AND	&	&	and, andi
Bitwise OR	|	|	or, ori
Bitwise NOT	~	~	nor

[group 10]

8. Question:

MIPS doesn't provide branch instructions such as blt,bgt,ble,bge. But we can use instruction such as beq, bne, slt in MIPS to execute the above four instructions.
Please match(A)to(D)with following instruction sequence(1)to(4)
(A) blt $s1,$s2,L (B) bgt $s1,$s2,L
(C) ble $s1,$s2,L (D) bge $s1,$s2,L

(1)
slt $t0, $s1, $s2
bne $t0, $zero, L
(2)

slt $t0, $s2, $s1
beq $t0, $zero, L
(3)
slt $t0, $s1, $s2
beq $t0, $zero, L
(4)
slt $t0, $s2, $s1
bne $t0, $zero, L

Ans:

(1)A (2)C (3)D (4)B

[group 12]

9. Question:

In MIPS, the J-type instruction "j label" is used for unconditional jumps. How can we achieve the same effect using an I-type instruction? Provide the MIPS instruction and explain its mechanism.

Ans:

beq $0, $0, label
Explanation:
(* beq compares two registers and branches if they're equal.
* $0 is the zero register, always containing 0.)
* beq $0, $0, label always evaluates as true, causing an unconditional branch to the specified label.

[group 13]

10. Question:

為什麼 mips 中沒有 shift left arithmetic (sla)指令？

Ans:

因為 mips 的設計主要是為了簡化設計和效率，需要簡單高效，但 mips 已經提供了 shift left logical (SLL) 來左移，大部分情況已經夠用了，並且 sla 因為會改變符號位，因此其實不是很常用。

[group 14]

11. Question:

Assume that the opcodes of the add and lw instructions are respectively 000000 and 100011 in binary notation. Translate the following MIPS machine codes into corresponding assembly codes.

(1) 00000001001010100100000000100000

(2) 10001101001010000000000000010100

Ans:

(1) add $t0, $t1, $t2

(2) lw $t0, 20($t1)