Computer Architecture

Fall, 2020

Week 4

2020.10.5

Group:

組員簽名：________________________________

[group5] (對抗賽)

1. Which following description is correct？
   - (A) The benefit of using registers is that it has unlimited number of storage elements built directly in the processor.
   - (B) (In MIPS) Translate the following statement into assembly code: f = g -10. The answer will be: subi $s0, $s1, 10
   - (C) (In MIPS)In the operation of hardware, each instruction is 32 bytes.
   - (D) Registers in hardware are faster than memory.
   - (E) We put 'typical constants' in memory since memory is more versatile.

Ans : (D)

a. "unlimited" => "limited" (and it's not a benefit too)

b. "subi $s0, $s1, 10" => "addi $s0, $s1, -10"

c. "bytes" => "bits"

e. The registers are more versatile, 且前後敘述無關連性。

[group6] (對抗賽)

2. C code: A[10]=g+A[4] for which g in $s2,base address of A in $s3. Which steps are correct, find and rearrange them.
   - (A) sw $t0,16($s3)
   - (B) sw $t0,40($s3)
   - (C) add $t0,$s2,$t0
   - (D) sub $t0,$s2,$t0
   - (E) lw $t0,16($s3)
   - (F) lw $t0,40($s3)

Ans : (E)(C)(B)

```
lw $t0,16($s3)
add $t0,$s2,$t0
sw $t0,40($s3)
```

[group7] (對抗賽)

3. In MIPS ISA, please choose the wrong statements and explain your answer.
   (A) Each instruction is 32 bits.
   (B) MIPS arithmetic instructions can be performed on register and memory.
   (C) 1 word = 4 bytes = 32 bits
   (D) Each memory address is 32 bits
   (E) We can write a value into any register.

Ans : (B)(E)

(B) 不再 memory 執行，要把 memory 的值 load 到 register 才能執行

(E) MIPS register 0 ($zero) is the constant 0. It cannot be overwritten.

[group8] (對抗賽)

4. Please tell the following statements are true or false. If false, please also tell us why.
   (A) Memory in hardware is faster than registers.
   (B) Although registers are faster than memory, we still might keep variables in memory.
   (C) This is defined in hardware, so instruction like add $t2, $zero,$s1 will not do anything.
   (D) In MIPS instructions syntax is rigid to 1 operands and 3 operator
   (E) If the ISA is MIPS , we can directly specify memory's address when we do data transfer.
   (F) In MIPS , the instruction set of control transfer contain load/store, computational and memory management.
   (G) We can't put the data to any address in the memory.

Ans :

(A) F, Memory in hardware is slower than registers

(B) T

(C) F, instruction add $t2, $zero,$s1 means that move data in register $s1(variable) to register $t2(temporary)

(D) F, In MIPS instructions syntax is rigid to 3 operands and 1 operator

(E) F, We need to use point and offsets.

(F) F, Memory management is not the instruction sets of control transfer.

(G) T

[group14]

5. Choose the incorrect MIPS instruction.
   (a). lw $s1 , 20($s2)
   (b). sw $s2 , 15($s3)
   (c). addi $s3 , $s4 , 5
   (d). sub $s5 , $s6 , $s7

Ans : (b), 15 is not a multiple of 4

[group3] (對抗賽)

6. Show how the value 0xfedcba09 would be arranged in memory of a little-endian and a big-endian machine. Assume the data is stored starting at address 100.

Big-Endian		Little-Endian	
Address	Data	Address	Data
100		100	

Ans

Big-Endian		Little-Endian	
Address	Data	Address	Data
100	0xfe	100	0x09
101	0xdc	101	0xba
102	0xba	102	0xdc
103	0x09	103	0xfe

[group4] (對抗賽)

7. Explain why simplicity favors regularity, and why it is a good design principle.

Ans

"Simplicity favors regularity" means that regularity in hardware design makes implementation simpler. Regular hardware designs are easier to implement and work with because they adhere to a standardized set of rules and guidelines. Simple hardware is good design because it enables higher performance at lower cost.

[group9]

8. Compile the C code: A[24] = B + A[4]; B in $s1, base address of A in $s2, then we can get a compiled MIPS code:

```
lw $t0, X($s2)
add $t0, $sY, $t0
sw $t0, Z($s2)
```

What is the value of "Z/X+Y"?

Ans: 7 (X=16, Y=1; Z=96)

[group13] (對抗賽)

9. Suppose there are three arrays, which are A, B, C, and their base addresses are stored in register $s3, $s4, and $s5 respectively. In addition, two variables j, and k are stored in register $s0, $s1 respectively. The current temporary register is $t0.

   Consider the following instructions:

   ```
   A[1] = k + (j + B[3])
   A[4] = A[1] + 10
   C[5] = A[4] – 2
   ```

   And its corresponding MIPS assembly code is:

   ```
   lw $t0, 12($s4)
   add $t1, $s0, $t0
   add $t0, $s1, $t1
   sw $t0, 4($s3)
   lw $t0, 4($s3)
   add $t1, $t0, 10
   sw $t1, 4($s3)
   lw $t0, 4($s3)
   sub $t1, $t0, 2
   sw $t1, 20($s5)
   ```

   Are there any mistakes in the above code? If there is, point it out and correct it.

Ans

```
lw $t0, 12($s4)
add $t1, $s0, $t0
add $t0, $s1, $t1
sw $t0, 4($s3)
lw $t0, 4($s3)
add $t1, $t0, 10 * // It should be addi
sw $t1, 4($s3) * // The correct offset is 16($s3)
lw $t0, 4($s3) * // Same as above
sub $t1, $t0, 2 * // The correct code should be addi $t1, $t0, -2
sw $t1, 20($s5)
```