Computer Architecture

Fall, 2020

Week 5

2020.10.12

組別：_____ 簽名：_____

[group1]

1. $t0 = 0xBEADFEED, $t1 = 0xDEADFADE What is the value of $t2 for the following instructions?

```
(1) sll $t2, $t0, 4; andi $t2, $t2, -1
(2) srl $t2, $t0, 3; andi $t2, $t2, 0xFFEF
```

Ans:

(1) 0x0000EED0 (2) 0x0000BFCD

[group11]

2. About the 32-bit MIPS instructions, which description is correct?

(a) MIPS has 32 registers inside CPU because it is a 32-bit CPU.

(b) add instruction can not directly store the addition result to memory.

(c) since memory structure is byte-addressing, the address offset in beq instruction is referred to as byte.

(d) In MIPS, “branch-fi-less-than” is realized using slt and beq/bne, since its design, principle is two faster instructions are more useful than one slow and complicated instruction.

A: b

(a): 32-bit CPU means the length of register is 32

(c): address offset in beq instruction isreferred to as word

(d): design principle - smaller is faster (keeping instruction set small)

[group12] (對抗賽)

3. Suppose we have a 32-bit computer with an instruction set that supports immediate instructions as shown below:

Opcode	Source Register	Destination Register	Immediate
6 bits	5 bits	5 bits	16 bits

(1) How many registers at most does this computer have?
(2) How many operations at most can this computer have?
(3) What is the range of the number in the Immediate field in 2's complement format?

Answer:
(1) $2^5 = 32$ registers
(2) $2^6 = 64$ registers
(3) $-2^{15} \sim +2^{15} - 1 = -32768 \sim +32767$

[group13] (對抗賽)

4. Show the result of $t2 in hexadecimal notation after doing the following MIPS instructions, where $t0=0x9487bace, $t1=0xbed87453.

```
sll $t2 $t0 2
addi $t1 $t1 -1
and $t2 $t1 $t2
srl $t2 $t2 4
ori $t3 $t2 0xab
nor $t2 $t3 $zero
```

Ans:
sll $t2 $t0 2 //$t0 shift left by 2 bits -> $t2
addi $t1 $t1 -1 //$t1-1 -> $t1
and $t2 $t1 $t2 //$t1 & $t2 -> $t2
srl $t2 $t2 4 //$t2 shift right by 4 bits -> $t2
ori $t3 $t2 0xab //$t2 or 0xab -> $t3
nor $t2 $t3 $zero //$~t3 -> $t2(or $t2=~(t3 | zero))
=> $t2=0xfede7954
**詳解

Please find the result of $t2 after doing the following MIPS instructions

$t0 = 0x9487bace

$t1 = 0xbed87453

a	10
b	11
c	12
d	13
e	14
f	15

① sll $t2 $t0 2 // $t0 shift left by 2 bits → $t2

② addi $t1 $t1 -1 // $t1-1 → $t1

③ and $t2 $t1 $t2 // $t1 & $t2 → $t2

④ srl $t2 $t2 4 // $t2 shift right by 4 bits → $t2

⑤ ori $t3 $t2 0xab // $t2 or 0xab → $t3

⑥ nor $t2 $t3 $zero // ~ $t3 → $t2

$t0:	9	4	8	7	b	a	c	e
	1001	0100	1000	0111	1011	1010	1100	1110
	b	e	d	8	7	4	5	3
$t1:	1011	1110	1101	1000	0111	0100	0101	0011
①	0101	0010	0001	1110	1110	1011	0011	1000 → $t2
②	1011	1110	1101	1000	0111	0100	0101	0010 → $t1
③	0001	0010	0001	1000	0110	0000	0001	0000 → $t2
④	0000	0001	0010	0001	1000	0110	0000	0001 → $t2
0xab	0000	0000	0000	0000	0000	0000	1010	1011
⑤	0000	0001	0010	0001	1000	0110	1010	1011 → $t3
⑥	1111	1110	1101	1110	0111	1001	0101	0100 → $t2
	f	e	d	e	7	9	5	4

⇒ $t2 = 0xfede7954

[group10] (對抗賽)

5. Which of the following statements about Instruction format are correct?

(A) MIPS defines only 3 basic types of instruction formats because of simplicity.

(B) In I format instructions like lw and sw, rs field always stands for base register and rt field always stands for destination register.

(C) add $t0 $s1 $s2 can be represented as 000000 10010 10001 01000 00000 100000. ($s1 = $17, $s2 = 18 $t0 = $8, function code of add = 100000)

(D) The opcode of R type instructions is always 000000, like add, sub, sll instructions.

(E) Because of the design principle "Smaller is faster.", the immediate field in I format instruction contains only 16 bits rather than 32 bits.

Ans:

(A) True

(B) False No, in sw instruction rt stands for source register

(C) False 000000 10001 10010 01000 00000 100000 (rs and rt field)

(D) True

(E) False "Smaller is faster." should be "Good design demands good compromises"

[group5] (對抗賽)

6. Please indicate true or false of the following statements and explain why.

A) No matter where data are (in registers or memory), instructions can directly operate on them.

B) The maximum and minimum immediate that can be implemented in I-Format are $\pm 2^{16}$

C) Use beq and bne instead of blt, bge, etc is an example of good design compromise.

D) We can write any value into any register.

Ans :

A) F, 在 memory 的要先移到 register

B) F, 只有 16bits,要 signed integer 只有$\pm 2^{16}$

C) T

D) F, $zero 無法被寫入

[group8] (對抗賽)

7. Please tell the following statements are true or false, if false, please also tell us why.

(a). In C, we have “ a* =8 ” which would compile to “ sll $s0,$s0,8 ” in MIPS.
(b). In R-Format Instructions, each register field is exactly 5 bits, which means that it can specify any unsigned integers in the range 0-31.
(c). One of the benefits for storing program computers is make people can be easily to operate on program.
(d). Binary compatibility allows compiled programs to work on different computers.
(e). MIPS has the shift instructions:SLA.
(f). The reason why we use shifting to multiply is that it's cheaper and faster.

ANS:
(a). false, it should be “sll $s0,$s0,3”
(e). false, we have ssl, srl, sra ,but not have sla
(b)(c)(d)(f). true

[group3] (對抗賽)

8. Provided fields of a MIPS instruction shown below. What is the type of this MIPS instruction? What is the assembly language form of this instruction? What is the binary representation of this instruction?

op=0, rs=18, rt=19, rd=9, shamt=0, funct=32

Ans:
(4) R-type
(5) Add $t1 $s2 $s3
(6) 000000 10010 10011 01001 00000 100000

[group4] (對抗賽)

9. In the online course, only the condition “branch on less than” was mentioned:

```
if (g < h) goto Less; // g = $s0, h = $s1
```

The statement can be present as the following assembly:

```
slt $t0,$s0,$s1
bne $t0,$0,Less
```

The instruction “Set on Less Than (slt)” and “Branch on Equal (beq)” was used to perform this operation. What about the condition “branch on less than or equals to”? Please compile the following statement in the most efficient way by hand:

```
if (g <= h) goto Less; // g = $s0, h = $s1
```

ANS:

slt $t0,$s1,$s0 # !(g <= h) ---> g > h

beq $t0,$0,Less