Computer Organization & Assembly Language

Midterm Exam – 2010/11/19

Dept. of Engineering Science, National Cheng Kung University

1. Assume using the MIPS architecture. Answer the following questions. (18%)

32 (A) How many bits are in a word?

8 (B) How many bits are used to represent a "char" in C?

64 (C) How many bits are used to represent a "double" in C?

2^{30} (D) How many words of memory are addressable by the MIPS?

ISA (E) One of the most important examples of an abstraction is the interface between hardware and the lowest-level software. What is this abstraction called?

(F) List the five class components of a computer as defined by the textbook authors.

Input output memory control datapath

2. Answer "True" or "False" to the following statements. (18%)

F (A) Assembly language pseudo-operations are executable.

F (B) The stack in MIPS grows from lower address to higher address.

T (C) Conditional branches in MIPS employ PC-relative addressing.

F (D) J-format (or J-type) instructions support conditional branching.

T (E) A left shift instruction can replace an integer multiply by a power of 2 if no overflow occurs.

T (F) The jump-and-link instruction (`jal`) in MIPS forms a link to the calling site by storing the address of next instruction in the register `$ra`.

3. Fill in the following blanks. (9%)

R-format:

op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

I-format:

op	rs	rt	Addr/Immediate
6 bits	5 bits	5 bits	16 bits

J-format:

op	address
6 bits	26 bits

4. Suppose die area is 0.4 cm^2 and there are 4 defects per cm^2. Calculate the yield. Then calculate the yield if defects per area can be cut in half. Note that the answers should be in the form of xx.xx%. (8%)

$$Yield = \frac{1}{\left(1 + DefectsPerArea \bullet \frac{DieArea}{2}\right)^2}$$

n=4 → n=3 → n=2 → n=1

5. Here is a mystery function. It expects one non-negative integer argument and returns one integer result.

```
yuck:
    li      $t0, 1          int i=1;
    li      $t1, 1          int j=1;
loop:
    blt     $a0, 2, exit    while (n>=2)
    add     $t2, $t0, $t1   { k=i+j;
    move    $t0, $t1          i=j;
    move    $t1, $t2          j=k;
    addi    $a0, $a0, -1      n=n-1;}
    j       loop            return j;
exit:
    move    $v0, $t1        add $v0 $t1 $zero
    jr      $ra
```

(A) Translate this yuck function into C. Assume that the usage of registers is known as (n: $a0 & i, j, k: $t0~$t2) where n, i, j, k are all C variables. (12%)

(B) What will this function return if it is called with an argument of 4 (i.e, $a0=4)? (4%) 5

6. Answer the following arithmetic questions. (16%)

(A) Convert -28410_{ten} to two's complement. Show the 32-bit binary representation.

(B) Convert 0xCA50 to binary.

(C) Convert 0xFFF7 to decimal, interpreting it as a signed 16-bit integer.

(D) Subtract 0x001F from 0xFFF7, interpreting each as 2's complement 16-bit integer. Express the answer both in hex and decimal.

7. Execute the following MIPS code fragments, showing the changes that occur in the register file and in memory. You only need to show the changes. (10%)

(A)
```
addi    $19, $0, 0x20
lw      $17, 0x04($19)
add     $20, $19, $16
sw      $20, 0x08($19)
```

(B)
```
addi    $1, $0, 0x20
lw      $2, 0x04($3)
add     $0, $3, $1
bne     $0, $1, loop
```

(C) Is the branch taken? (answer YES or NO)

Registers	BEFORE	AFTER	Memory	BEFORE	AFTER
$16	0x10	×	0x20	0x22	×
$17	0x14		0x24	0x30	×
$18	0x16	×	0x28	0x40	
$19	0x28		0x2C	0x50	×
$20	0x1234		0x30	0x60	×

Registers	BEFORE	AFTER	Memory	BEFORE	AFTER
$0	0x00	×	0x20	0x10	×
$1	0x14		0x24	0x30	×
$2	0x16		0x28	0x40	×
$3	0x28	×	0x2C	0x50	×
$4	0x1234	×	0x30	0x60	×

8. Fill in the missing line of MIPS assembly code so that it branches to the instruction labeled loop when the contents of the $t0 register are nonzero. (5%)

```
loop:
    lbu     $t0,0($s0)      # get byte pointed to by $s0
    addi    $s0,$s0,1       # increment pointer to next byte
    andi    $t0,$t0,127     # mask off most significant bit
    bne     $t0,$zero,loop
    sub     $s2,$s0,$s1     # subtract end pointer from start
    ...
```