Computer Organization & Assembly Languages

Midterm Exam - 2005/11/30

Dept. of Engineering Science, National Cheng Kung University

1. [24 points] General Knowledge

(A) Short Answer: Assume using the MIPS architecture.

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

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

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

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

(5) One of the most important examples of an abstraction is the interface between hardware and the lowest-level software. What is this abstraction called? 指令集架構

+4 ✓ (6) List the five class components of a computer as defined by the textbook authors.

(B) True/False: Write your response. Ambiguous responses will be marked wrong.

(7) True/False Assembly language pseudo-operations are executable.

(8) True/False The stack in MIPS grows from lower address to higher address.

(9) True/False Conditionai branches in MIPS employ PC-relative addressing.

(10) True/False J-format instructions support conditional branching.

(11) True/False The `lw` instruction assembles into two machine code instructions.

(12) True/False 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`.

2. [6 points] Fill in the following blanks.

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

3. [5 points] 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%.

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

4. [15 points] Translating MIPS code

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

```
yuck:
    li      $t0, 1
    li      $t1, 1
loop:
    blt     $a0, 2, exit
    add     $t2, $t0, $t1
    move    $t0, $t1
    move    $t1, $t2
    addi    $a0, $a0, -1
    j       loop
exit:
    move    $v0, $t1
    jr      $ra
```

(1) 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 points)

(2) What will this function return if it is called with an argument of 4 (i.e, $a0=4)? (3 points)

5. [15 points] Arithmetic

(1) Convert -28410 to two's complement. Show the 32-bit binary representation.

(2) Convert 0xCA50 to binary.

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

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

(5) Show the IEEE 754 representation of -1.25_{ten} in single precision.

6. [8 points] 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.

(1)

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

Registers	BEFORE	AFTER
S16	0x10	
S17	0x14	
S18	0x16	
S19	0x20	
S20	0x1234	

Memory	BEFORE	AFTER
0x20	0x22	
0x24	0x30	
0x28	0x40	
0x2C	0x50	
0x30	0x60	

(2)

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

Registers	BEFORE	AFTER
S0	0x00	
S1	0x14	
S2	0x16	
S3	0x28	x
S4	0x1234	

Memory	BEFORE	AFTER
0x20	0x10	x
0x24	0x30	x
0x28	0x40	x
0x2C	0x50	x
0x30	0x60	x

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

7. [5 points] Branch offset computation problem

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.

```
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
    ____________________
    sub     $s2,$s0,$s1     # subtract end pointer from start
```

8. [12 points] Compile the following C program into MIPS instructions. Assume that the usage of registers is specified as (g, h, i, j: $a0~$a3 and f: $s0).

```
int leaf_example (int g, int h, int i, int j){
    int f;
    f = (g+h) - (i+j);
    return f;
}
```

```
leaf_example:
    addi    $sp, $sp, -4
    sw      $s0, 0($sp)
    add     $t0, $a0, $a1
    add     $t1, $a2, $a3
    add     $s0, $t0, $t1
    move    $v0, $s0
    lw      $s0, 0($sp)
    jr      $ra
```

```
leaf_example:
    addi    $sp, $sp, -4
    sw      $s0, 0($sp)
    add     $t0, $a0, $a1
    add     $t1, $a2, $a3
    add     $s0, $t0, $t1
    move    $v0, $s0
    lw      $s0, 0($sp)
    jr      $ra
```

9. [10 points] Here is a C fragment that repeatedly calls sequence:

```
void main(){
    int t;
    int sum = 0;
    for(t = 0; t < 5; t++)
        sum = sum + sequence(t);
}
```

This might translate into the following assembly code:

```
main:
    sw      $ra, 0($sp)
    addi    $sp, $sp, -4
    # Initialize sum = 0 and t = 0
    add     $s0, $0, $0   # $s0 = sum
    addi    $t0, $0, 0    # $t0 = 0
    # Call sequence(t)
loop:
    move    $a0, $t0
    jal     sequence
    # Increment loop index t
    addi    $t0, $t0, 1
    # Compute sum = sum + sequence(t)
    add     $s0, $s0, $v0
    # Repeat while (t < 5)
    slti    $t1, $t0, 5
    bne     $t1, $0, loop
    addi    $sp, $sp, 4
    lw      $ra, 0($sp)
    jr      $ra
```

This code fragment does not correctly follow the MIPS *procedure calling convention.* What is wrong? Show the changes required to make this code observe the calling convention.