Name: . . . .

ID Number: . . . .

COMP203: Mid-Term Test

Model Solutions

04 April, 2007

Instructions

- Maximum time: **90 minutes**.
- Answer **all** the questions.
- There are 90 marks in total.
- Write your answers in the boxes in this test paper and hand in all sheets.
- Paper translation dictionaries are allowed.
- Non-programmable calculators are allowed.
- Every box with a heavy outline requires an answer.
- Page 11 provides some commonly used MIPS instructions and registers for your reference.

Questions	**Marks**
1. Basic Concepts	[10]
2. Registers, Memory, and Big Constants	[10]
3. Decision Making	[10]
4. Addressing Modes and Instruction Formats	[10]
5. Number Conversion	[10]
6. Boolean Expression and Logic Gates	[10]
7. Multiplication	[10]
8. Overflow Detection and Manipulation	[10]
9. Procedures/Functions	[10]
Total Marks	[90]

Question 1. Basic Concepts

[10 marks]

(a) [2 marks] Briefly define the term *CPU* in the context of computer organisation.

CPU, or *Central Processor Unit*, is the active part of a computer, which executes the instructions of programs.

(b) [2 marks] Briefly define the term *Memory* in the context of computer organisation.

Locations that programs are stored while they are running.

(c) [2 marks] Briefly define the term *Assembler* in the context of computer organisation.

Program that translates a symbolic version of an instruction into its binary version.

(d) [4 marks] Is a *ROM* a combinational logic block or a sequential logic block? Justify your answer.

A ROM is a combinational logic block. Although called "read only memory", its outputs purely depend on its inputs — it does not contain any memory element.

Question 2. Registers, Memory and Big Constants [10 marks]

(a) [6 marks] Consider the following C statement:

```
A[25] = A[10] + j;
```

Assume that register `$s1` holds integer variable `j` and that register `$s0` holds the base address of the integer array `A`. Write a sequence of MIPS instructions that directly corresponds to this statement. Use temporary registers if necessary.

```
lw $t0, 40($s0)
add $t0, $t0, $s1
sw $t0, 100($s0)
```

(b) [4 marks] Consider the following sequence of MIPS instructions:

```
lui $t1, 0x0231
ori $t2, $t1, 0xa2c4
addi $t3, $t1, 0xa2c4
```

What values will be stored in registers `$t1, $t2, $t3` after the above instructions are executed?

```
$t1 = 0x0231 0000

$t2 = 0x0231 a2c4

$t3 = 0x0230 a2c4
```

Question 3. Decision Making [10 marks]

Consider the following C code segment:

```
if (x < 10)
   x = x + m;
else
   x = x - m;
x++;
```

Assume that the registers `$s0` and `$s1` hold the integer variables `x` and `m`, respectively.

Write a sequence of MIPS instructions that directly corresponds to this C code segment. Use temporary registers if necessary.

```
      slti $t0, $s0, 10
      beq $t0, $zero, else
      add $s0, $s0, $s1
      j exit
else: sub $s0, $s0, $s1
exit: addi $s0, $s0, 1
```

Question 4. Addressing Modes and Instruction Formats [10 marks]

Use the following sequence of MIPS instructions labelled as 1 to 9 to answer questions (a) and (b).

```
1          slt $t0, $s1, $s0
2          beq $t0, $zero, Else
3          sub $s1, $s1, $s0
4          add $s1, $s1, $s1
5          addi $s1, $s1, 1
6          j Exit
7    Else: lw $t0, 4($s4)
8          add $s1, $s1, $t0
9    Exit: or $s1, $s1, $t0
```

(a) [7 marks] For each of the above instructions labelled as `1, 2, 3, 5, 6, 7,` and `9`, state its addressing mode and instruction format.

label	addressing mode	instruction format
1	Register	R type
2	PC-relative	I type
3	Register	R type
5	Immediate/constant	I type
6	Pseudo-direct	J-type
7	Base	I type
9	Register	R type

(b) [3 marks]

Calculate the value of the branch relative address (*the offset in machine code*) of `Else` in Instruction 2 "`bne $t0, $zero, Else`". **Present the final result only** in the box below.

4

Question 5. Number Conversion [10 marks]

This question concerns different formats of numbers. **Write only the final answer into the boxes.**

(a) [3 marks] Convert the decimal number `-1023` into a 16-bit two's complement binary number.

1111 1100 0000 0001

(b) [3 marks] Convert the 16-bit two's complement binary number `1111 1000 0000 0000` into a decimal number.

-2048

(c) [4 marks] Show the IEEE 754 binary representation of the the decimal floating point number `-7.875` in single precision format.

1100 0000 1111 1100 0000 0000 0000 0000

Question 6. Boolean Expression and Logic Gates [10 marks]

Given the following truth table for a PLA (Programmable Logic Array), answer questions (a), (b) and (c):

Input			Output	
A	B	C	D	E
0	0	0	1	0
0	0	1	0	1
0	1	0	0	1
0	1	1	0	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	0

(a) [3 marks] Give a boolean expression for each of D and E based on the truth table.

$D = \overline{A} \cdot \overline{B} \cdot \overline{C} + A \cdot B \cdot C$

$E = \overline{A} \cdot \overline{B} \cdot C + \overline{A} \cdot B \cdot \overline{C} + A \cdot \overline{B} \cdot \overline{C}$

(b) [6 marks] Design a PLA (Programmable Logic Array) to implement the truth table based on the boolean expressions you gave in part (a).

(c) [2 marks] Calculate the size of the PLA.

$$3 \times 5 + 5 \times 2 = 25$$

Question 7. Multiplication

[10 marks]

Calculate the following multiplication using the Booth's algorithm:

`0110` $\times$ `1110`

Assume that the multiplicand and the multiplier are 4-bit 2's complement integers (consider the sign). Show your work in a table and identify your final result.

Loop	Action– steps	Mcand	Prod
0	Initialisation	0110	0000 1110 0
1	1c: no operation	0110	0000 1110 0
	2: Prod>>1	0110	**0000 0111 0**
2	1d:Prod=Prod-Mcand	0110	**1010** 0111 0
	2: Prod>>1	0110	**1101 001 1 1**
3	1d:no operation	0110	1101 0011 1
	2: Prod>>1	0110	**1110 1001 1**
4	1d: no operation	0110	1110 1001 1
	2: Prod>>1	0110	**1111 0100 1**

The final result is **1111 0100**.

Question 8. Overflow Detection and Manipulation [10 marks]

Assume that A and B are positive integers stored in registers `$s1` and `$s2`, respectively. Write a sequence of **at most 12** MIPS instructions to process all the following tasks:

- $C = A + B$; (Store C in register `$s3`)
- If there is no overflow, then subtract decimal constant 30 from C and place the result in register `$s4`;
- Otherwise, set the least significant bit of C to 1 and set the most significant bit of C to 0.

Use temporary registers if necessary.

```
            add $s3, $s1, $s2
            slt $t0, $s3, $zero
            bne $t0, $zero, Overflow
            addi $s4, $s3, -30
            j Exit
Overflow:   ori $s3, $s3, 0x0001
            lui $t1, 0x7fff
            ori $t1, $t1, 0xffff
            and $s3, $s3, $t1
Exit:
```

Question 9. Procedures/Functions [10 marks]

Given the following C procedure/function:

```
int test(int x, int y, int z)
{
   int w;

   w = (x + y) - (z - 2);

   return w;
}
```

Assume that the registers `$a0, $a1,` and `$a2` hold the parameters `x, y` and `z`, respectively, that register `$s1` holds the local variable `w`, and that both the caller and the callee need to use `$s1`. Write a sequence of MIPS instructions that directly corresponds to this function. Use temporary registers if necessary.

```
test:
  addi $sp, $sp, -4   # adjust stack
  sw $s1, 0($sp)      # push $s1
  add $t0, $a0, $a1   # $t0 = x + y
  addi $t1, $a2, -2   # $t1 = z - 2
  sub $s1, $t0, $t1
  add $v0, $s0, $0    # $v0 for result return
  lw $s1, 0($sp)      # restore $s1
  addi $sp, $sp, 4    # adjust stack, pop
  jr $ra              # return
```

A Commonly Used MIPS Instructions

add	sub
lw	sw
addi	lui
and	or
andi	ori
sll	srl
jal	jr
j	
beq	bne
slt	slti
mult	div
mul	
lb	sb

B MIPS Registers — Numbers and Names

Name	Number	Usage
$zero	0	constant value 0
$at	1	reserved for assembler
$v0–$v1	2–3	values for results and expression evaluation
$a0–$a3	4-7	arguments, for functions/procedures
$t0–$t7	8-15	temporaries
$s0–$s7	16-23	saved. Fast locations for data
$t8–$t9	24-25	more temporaries
$k0–$k1	26-27	reserved for the OS
$gp	28	global pointer
$sp	29	stack pointer
$fp	30	frame pointer
$ra	31	return address, for functions/procedures

****