UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE AND ENGINEERING
MIDTERM EXAMINATION, MARCH 2009

ECE243H1 S – COMPUTER ORGANIZATION
Exam Type: D
Duration: 2 Hours

A. Moshovos

This is a type D exam. You are allowed to use any printed material and a calculator as allowed by the University regulations.

Last Name (Print): _____

First Name: _____

Student Number: _____

	Marks	Max. Marks
1		10
2		10
3		15
4		20
5		15
6		20
7		10
Total		100

Please:
State your assumptions. Show your work. Comment your code. **Use your time wisely as not all questions will require the same amount of time.** If you think that assumptions must be made to answer a question, state them clearly. If there are multiple possibilities, comment that there are, explain why and then provide at least one possible answer and state the corresponding assumptions.

1. [10] Data Representation:

What is **`1111 0000`**?

a) Written in hexadecimal? 0xF0

b) If interpreted as an unsigned number? _240_____ (in decimal please)

c) If interpreted as a signed number in 2's complement? __-16____ (in decimal please)

d) If interpreted as a real number with fixed-point representation as 11.110000? Write the result in decimal: _3.75_____

e) Interpret the following 32-bit quantity as a single-precision floating point number in IEEE format:

`1100 0011 1000 0000 1100 0000 0000 0000`

First write the following in binary:

Sign bit = __1__

Exponent field = _1000 0111_____

Mantissa = _000 0000 1100 0000 0000 0000_____

Value in decimal = _-257.5_____

2. [10] Little vs. Big Endian: Assume that NIOS II is **big endian** for this question. Given the following initial state:

```
R4 =  0x12345678
R6 =  0xFABCFAAF
R7 =  0x56788765
```

Write the register values below after the following code executes:

```
stw  r6,  0(r4)
sth  r6,  4(r4)
sth  r7,  6(r4)
ldw  r10, 4(r4)
ldh  r11, 0(r4)
ldhu r12, 0(r4)
ldb  r13, 7(r4)
```

R10 = _0Xfaaf 8765____ R11 = 0XFFFF FABC_____

R12 = 0X0000 FABC __ R13 = 0X0000 0065

3. [15] Basic Instruction Semantics: Assume the following initial state for registers:

```
R4 =  0x12345678
R5 =  0xBADFADEB
R6 =  0xFEEDABBA
R7 =  0x56788765
PC =  0x00001000
```

And that starting at address 0x00001000 the following instructions appear:

```
     call foo
     addi r4, r4, -16
     ldw  r10, 16(r4)
     ldh  r11, 20(r4)
     ldhu r12, 18(r4)
     ldb  r13, 17(r4)
     andi r5, r5, 0xFFF1
     ori  r6, r0, 0x7FFF
     beq  r6, r10, skip
     andi r4, r4, 0xFF
skip: or  r0, r0, r0
     ...
foo: stw  r6,0(r4)
     sth  r6,4(r4)
     sth  r7,6(r4)
     ret
```

What are the following register values after the instruction at "skip", "or r0, r0, r0", has finished executing? Write the values in **hexadecimal, using all eight digits** (that is write 0x00000000 and not 0x0). If an instruction is not legal (i.e., the instruction does not exist, or the given use of operands is not allowed for the instruction), then write 'illegal' beside the instruction and proceed as if it had no effect.

R0 = _____ R4 = _____

R5 = _____ R6 = _____

R6 = _____ R7 = _____

R9 = _____ R10 = _____

R11 = _____ R12 = _____

R13 = _____ RA = _____

PC = _____

4. [20] Data Structure Representation and Manipulation:

a) Translate the following C declarations into assembly:

```
struct    ouf_t {
                int a;
                char b, c;
                int  d;
                } moo[2] = { {1, 30, 3, 4}, {5, 60, 7, 8} };
```

Complete the data section declaration below:

```
            .data
            .align 2
moo:        .long 1
            .byte 30,3
            .align 2
            .long 4
            .long 5
            .byte 60,7
            .align 2
            .long 8
```

b) If `i` is in `r8` complete the code below that reads into `r2` the value of **`moo[i].d`:**

```
            .text
            .align 2
            movia r9, moo
Got to calculate r9 + r8 * 12 + 8
```

5. [15] Machine Code and Functions:

a) Implement a function that takes a single argument (unsigned integer of 32-bits) and returns whether the number is not divisible by 2 nor divisible by 4. If the number is divisible by 2 or 4 the function should return zero, otherwise return a non-zero value.

```
    .text
isnd24:


    andi r2, r4, 1
    ret
```

b) Given:

```
    I1:     call I2
    I2:     addi ra, ra, 4
    I3:     ret
    I4:     or   r9, r9, r10
```

Show the sequence of instructions that will be executed as a comma separated list:

I1, I2, I3, I3, …

c) Is this an infinite loop?

```
     movi    r9, -1
loop: addi   r9, r9, r9
     ble     r9, r0, loop
```

What are the values that r9 takes during this loop? Does the loop terminate? Explain why.

R9 = -1 (0xFFFFFFFF), 0xFFFF FFFE, …, 0, 0, 0, 0, …

6. [20] Calling Conventions: Write the code for the following function:

```
int
zoom (int a, int b, int c)
{
   return b + broom (b + a, c, 1, 2, 3, 4);
}
```

```
        .text
zoom:   # prologue

      addi sp, sp, -16
      stw ra, 12(sp)
      stw r5, 8(sp)


      # pre-call for broom
      add r4, r4, r5
      add r5, r6, 0
      movi r6, 1
      movi r7, 2
      movi r2, 3
      stw r2, 4(sp)
      movi r2, 4
      stw r2, 0(sp)

      call broom
      # post-call

      # b + ...
      ldw r5, 8(sp)
      add r2, r2, r5



      # epilogue

      ldw ra, 12(sp)
      addi sp, sp, 16

      ret
```

7. [10] Machine Code: If initially we have:

```
R4 = 0x2
R5 = 0x1
R6 = 0x3
```

What are the values of the registers after the code below executes?

```
     .data
     .long     caseA + caseB, caseB, jt, 1000
jt:  .long     caseD, caseA, caseD, caseD, caseB + 8
     .long     caseC, caseA, caseE, caseA

     .text
     movia     r8, jt
     add       r9, r9, r9
     add       r9, r9, r9
     add       r8, r8, r9
     ldw       r8, 0(r8)
     jmp       r8                  # this does: PC = r8
caseA:
     add r4, r0, r0
     br after
caseB:
     add r5, r0, r0
     br after
caseC:
     add r5, r5, r4
     br after
caseD:
     add r6, r0, r0
     br after
caseE:
     add r6, r4, r5
after:
```

a) If initially r9 = 4: → CaseC= CaseB + 8 above gets executed

R4 = _____

R5 = _____ R6 = _____

b) If initially r9 = 3: → CaseD above gets executed

R4 = _____

R5 = _____ R6 = _____