Computer Science COMP-573A
Microcomputers
Fall 2003 Midterm
Student Solution Guide

Thursday, October 30th, 2003
10:00-11:30

- *Don't forget to write your name and your student id on your answer booklet.*
- *No calculator of any kind allowed.*
- *Grade for exams written in pencil are FINAL.*

Note that the solutions proposed in red are just one out of many possible solution. There may be other correct solutions.

Question 1: Data Representation (20 marks)

a) Let A = 0x97563252 and B = 0x79985611. For each of the following question, explain how you reached your conclusion.

(i) *(3 marks)* Which number is bigger if A and B are 32-bit integers in two's complement? Why?

B is greater than A. If we look at the first bit of A, we see the value 1. This generally indicates a negative number. The first bit of B is 0, so we have a positive number.

(ii) *(3 marks)* Which number is bigger if A and B are 32-bit unsigned integers? Why?

A is greater than B. With unsigned value, you can simply compare the two hex values.

(iii) *(3 marks)* Which number is bigger if A and B are IEEE single precision numbers? Why?

B is greater than A. With floating point numbers, the first bit tells us if a number is positive or negative. If we look at the first bit of A, we see a value of 1, so the number is negative. The first bit of B is 0, so we have a positive number.

b) What would be the content of the address A, A+1, A+2, A+3 if you store the 32-bit value 0x1234CAFE at address A on the following architecture (you can write your answer in hexadecimal format) :

(i) Intel *(little-endian) (3 marks)*

```
A      FE          or   1111 1110
A+1    AB               1010 1011
A+2    34               0011 0100
A+3    12               0001 0010
```

Or

```
A+3    12          or   0001 0010
A+2    34               0011 0100
A+1    AB               1010 1011
A      FE               1111 1110
```

(ii) Sparc *(big-endian) (3 marks)*

```
A      12          or   0001 0010
A+1    34               0011 0100
A+2    AB               1010 1011
A+3    FE               1111 1110
```

Or

```
A+3    FE          or   1111 1110
A+2    AB               1010 1011
A+1    34               0011 0100
A      12               0001 0010
```

c) Convert the following IEEE single precision floating-point number to a double precision floating-point number:

(i) 0x40000000 *(2 marks)*

In binary: 0100 0000 0000 0000 0000 0000 0000 0000
Sign bit: 0
Exponent: 1000 0000 : 128
Mantissa: 0000 0000 0000 0000 0000 000

Remove bias for single: 128 – 127 = 1
Add bias for double: 1 + 1023 = 1024

New Sign bit: 0
New Exponent: 100 0000 0000 : 1024
New Mantissa: 0000 0000 0000 0000 0000 0000 0000 0000 …

Double Precision: 0100 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
Or Hex: 0x4000 0000 0000 0000

(ii) 0x79985611 *(3 marks)*

In binary: 0111 1001 1001 1000 0101 0110 0001 0001
Sign bit: 0
Exponent: 111 1001 1: 243
Mantissa: 001 1000 0101 0110 0001 0001

Remove bias for single: 243 – 127 = 116
Add bias for double: 116 + 1023 = 1139

New Sign bit: 0
New Exponent: 100 0111 0011: 1024
New Mantissa: 0011 0000 1010 1100 0010 0010 0000 0000 0000 0000 …

Double Precision: 0100 0111 0011 0011 0000 1010 1100 0010 0010 0000 0000 0000 0000 0000 0000 0000
Or Hex: 0x4733 0AC2 2000 0000

Question 2 : Intel Architecture (22 marks)

a) *(10 marks)* Write the Intel x86 assembly code that would be generated for this C program. Document your assembly program with the C code statements and comments.

```
int thefunction (unsigned int n)
{
        int i;

        i = 0;

        while ( n != 0) {

                if ( (n & 1) == 1 ) i++;
                n = n / 2;

        }

        return i;
}
```

Example of a solution:

```
        .text
.globl thefunction
        .type thefunction, @function
thefunction:
        pushl %ebp
        movl %esp, %ebp
        subl $4, %esp
        movl $0, -4(%ebp)
.Start_While:
        cmpl $0, 8(%ebp)
        je .End_While
        movl 8(%ebp), %eax
        andl $1, %eax
        cmpl $0, %eax
        je .After_If
        incl -4(%ebp)
.After_If:
        movl 8(%ebp), %eax
        shrl %eax
        movl %eax, 8(%ebp)
        jmp .Start_While
.End_While:
        movl -4(%ebp), %eax
        leave
        ret
```

b) *(2 marks)* What does this function do? *(This answer should only require a sentence or two.)*

It counts the number of 1 in the binary representation of the number.

c) *(4 marks)* What are the 4 data registers available to the programmer? What are they used for?

From the floppy text book:

EAX: The Accumulator must be used for a few arithmetic instructions, such as MUL and DIV; it is also used for I/O and many instructions perform more efficiently if they use EAX, AX or AL rather than any other register.
EBX: The Base register is the only one of these four that can be used to index into memory; EBX normally points to the Data Segment.
ECX: The Counter is normally used to control the execution of loops. As we will see later, ECX is automatically decremented by special loop and string instructions. ECX is also used to shift and rotate by more than one bit at a time.
EDX: The Data register is used by a few instructions to extend the Accumulator to 64 bits.

Note that simply giving the name of the register was worth no point.

d) *(6 marks)* Name and explain three improvements that were added to the original x86 design to produce today's modern Pentium processor? *(Explanations should be short and concise.)*

The list of possible answer is quite extensive:

- *Protected mode operations (descriptor tables)*
- *Virtual memory management*
- *New protection mecanism (segment limit checking, etc)*
- *32-bit processor*
- *Flat memory model*
- *Paging (fixed 4-Kb page size)*
- *Limited parallele stages (6 stages)*
- *Improved parallele execution*
- *Integrated FPU (Floating Point Unit)*
- *On-chip level 1 cache*
- *Power management (late in the series)*
- *Addition of another execution pipeline (making the architecture superscaler)*
- *Branch prediction*
- *Built-in multi-processor support*
- *MMX (Multimedia Extensions)*
- *Dynamic Execution (out-of-order execution, superior branch prediction, flow*
- *analysis, etc)*
- *On-chip level 2 cache*
- *Improved power management (sleep, etc)*
- *Streaming SIMD Extensions (SSE)*
- *NetBurst micro-architecture*
- *Rapid Execution Engine*
- *Hyper Pipeline Technology*
- *Streaming SIMD Extensions 2 (SSE2)*

Question 3 : Sparc Architecture (14 marks)

At the beginning of a SPARC assembly language function, one should use the following instruction:

```
save    %sp, -WINDOWSIZE, %p
```

a) *(4 marks)* Explain the meaning of this instruction?

This instruction it used to rotate the current register window and allocate a new stack frame.

b) *(4 marks)* What is the minimum value of WINDOWSIZE? Why?

*The minimum value of the window size is 92 (16*4 for register dumping + 1*4 for hidden return value + 6 * 4 for six first argument).*

96 is also a valid answer because window sizes must be written in a multiple of 8.

Some student also mentioned the extra 20 bytes GCC reserves for itself. Thought this space is not officially in the stack frame, I've decided to accept it as a valid answer. As such, the value 112 would also be a good answer.

c) *(3 marks)* Why would you need a larger value?

- *Needed temporary space*
- *Needed space for local variable*
- *Calling function with more than 6 arguments*

d) *(3 marks)* When do you NOT need to use this instruction?

- *Did not need additional registers*
- *Did not need stack window*
- *Did not need local variables*

Question 4 : Power PC Architecture (14 marks)

Translate the following PowerPC assembly function into C code.

```
        .section        ".text"
        .align 2
        .globl func
        .type   func,@function
func:
        stwu 1,-48(1)
        stw 31,44(1)
        mr 31,1
        stfs 1,8(31)
        stw 3,12(31)
        stfs 2,16(31)
        lis 0,0x4000
        stw 0,24(31)
        lwz 0,12(31)
        stw 0,20(31)
.L2:
        lwz 0,20(31)
        cmpwi 0,0,0
        bgt 0,.L5
        b .L3
.L5:
        lfs 13,24(31)
        lfs 0,8(31)
        fmuls 0,13,0
        stfs 0,24(31)
        lwz 9,20(31)
        addi 0,9,-1
        stw 0,20(31)
        b .L2
.L3:
        lfs 13,24(31)
        lfs 0,16(31)
        fdivs 0,13,0
        stfs 0,24(31)
        lfs 0,24(31)
        fmr 1,0
        lwz 11,0(1)
        lwz 31,-4(11)
        mr 1,11
        blr
```

Hint: The function header is “float func (float x, int y, float z)”

Here is the original C code used to generate the assembly code:

```
float func ( float x, int y, float z) {

        int i;
        float temp;

        temp = 2.0;

        for ( i = y; i > 0; i-- ) {
                temp = temp * x;
        }

        temp = temp / z;

        return temp;
}
```

Question 5 : Floating Point Arithmetic (16 marks)

Given the following expression:

$$\sqrt{\frac{a^2 + 2*b + c}{a + c}}$$

The variables *a*, *b* and *c* are floating-point numbers. You can assume that their values are respectively stored in the third, fourth and fifth floating point register.

Write the assembly code to evaluate the given mathematical expression in the following architecture:

Note: During the exam, I realized (with student help) that the problem could be interpreted in two ways. A such, the value a,b,c can be found in register r2, r3, r4 or r1, r2, r3. Both interpretations are correct and not points were taken away as long as the student was coherent in their interpretation.

a) *(7 marks)* in Sparc assembly (in single-precision).

Here is an example solution:

```
fmuls %f2, %f2, %f5
fadds %f3, %f3, %f6
fadds %f5, %f6, %f5
fadds %f5, %f4, %f5
fadds %f2, %f4, %f6
fdivs %f5, %f6, %f5
fsqrts %f5, %f2
```

b) *(7 marks)* in PowerPC assembly (in double-precision).

Here is an example solution:

```
fmul 13,2,2
fadd 12,3,3
fadd 12,12,13
fadd 12, 12, 4
fadd 13, 2, 4
fdiv 12,12,13
fsqrt 2,12
```

Note:

- *The answer should be place in the third floating-point register*
- *You do not need to worry about overflow or underflow*
- *Feel free to comment you code so we can better understand*

Question 6 : Other Architectures (14 marks)

a) *(7 marks)* The ARM instruction set has 25 conditional instructions. Why and how do they work?

Every instruction in the instruction set is condition. The instruction format includes 4-bit that allow for every instruction to have it's own condition code.

b) *(7 marks)* The Intel Itanium features powerful branch prediction mechanisms. To reduce dependencies between different instructions, the architecture features an alternative branching system. This allows any instruction to be conditional. What is this system and how does it work?

The Itanium features predicate, 1-bit registers appended to every instruction. A typical Itanium processor has 64 of these 1-bit predicate registers. When an instruction is executed, at the last stage of the pipeline, the predicate bit is checked. If the value of the predicate register is 0, then the instruction is cancelled. Otherwise, the instruction finishes uninterrupted. This decreases dependencies between instructions.