University of Victoria
Final Exam
April 2010

Course	: Computer Science 230
Section	: A01
CRN	: 20853
Instructor	: Jason Corless
Duration	: 3 hours

This exam has a total of 6 pages including this cover page.

Students must count the number of pages and report any discrepancy immediately to the Invigilator.

This exam is to be answered in the booklets provided.

The total number of marks in this exam is 32.

In this exam:

0x123 is used to indicate the number 123 in base 16.

Question 1 – 4 marks

Perform the following conversions. Show your work. 1 mark each.

a) 1101010 base 2 to base 10.

b) 03D7 base 16 to base 2

c) 0.543 base 10 to base 2. Provide a maximum of 4 bits of precision.

d) -13 base 10 to 10 bit two's complement. Show your result in base 2.

Question 2 – 2 marks

Draw a timing diagram that shows an output handshaking protocol that uses a single line that is asserted when the interface wants to send data, a data port (or data bus) and a single line that is used by the device to acknowledge receipt of the data.

Include an indication as to (i) when in the timing diagram the data on the data port is valid and (ii) when in the timing diagram the device accepts the data.

Question 3 – 2 marks

Consider the following ARM assembly program:

```
          .text
          .global   _start

_start:        mov  r0,#2
               mov  r1,#4
               mov  r3,#1
               movs r2,r1
loop:          beq  done
               mul  r4,r3,r0
               mov  r3,r4
               subs r2,r2,#1
               b    loop
done:          swi  0x11
```

a) What is the value of r3 when the swi instruction is executed?
b) Considering r0 and r1 as inputs into this program, what is this program calculating?

Question 4 – 4 marks

Write a complete function called `array_copy` in ARM assembly language that takes 4 parameters:

r1 – address of the input array

r2 – address of the output array

r3 – size of the input array

r4 – threshold value

The function should copy all values in the input array that are less than the threshold value into the output array. The input and output arrays contain words (ie. 4 byte values). You may assume there is enough space in the output array to hold all possible values.

On return from your function, r3 should contain the number of values copied into the output array.

Be sure to save and restore any registers you modify (including r1, r2, and r4).

Question 5 – 4 marks

Consider a hypothetical system with a 2^{10} word addressable memory space that is divided into 2^8 blocks, with 4 words per block. For this question, assume we have a cache with 2^4 slots.

a) For this system with a direct mapped cache, which slot would a reference to the address 0x057 be mapped to? What is the tag value for this address?

b) For this system with a direct mapped cache, what pattern of memory access would result in the same slot being repeatedly filled with different blocks therefore causing a sequence of cache misses?

c) Now consider this system with a two-way set associative cache. How many sets does the cache have?

d) Again considering a two-way set associative cache, which set would a reference to the address 0x123 be mapped to? What is the tag value for this address?

Question 6 – 4 marks

Throughout the term we made use of the SWI instruction.

First, consider the following opcodes for some SWI instructions:

Instruction	Opcode
swi 0x00	0xEF000000
swi 0x01	0xEF000001
swi 0xF2	0xEF0000F2
swi 0x03	0xEF000003

The important thing to note is that the particular SWI routine that is being requested is encoded in the lower 3 bytes of the SWI opcode.

When the processor encounters an SWI instruction, it transfers control to the SWI handler vector at memory location 0x08. (There is also a mode switch, but we will ignore that.)

In addition, r14 contains the address of the instruction after the SWI instruction. That is, r14 contains the return address.

a) If your SWI handler routine is called `swi_handler`, what instruction must be placed at memory location 0x08?

b) When executing inside the `swi_handler`, r14 contains the address of the instruction after the swi instruction that caused control to transfer to the `swi_handler`. Write a sequence of instructions that extracts the swi routine number to be invoked into register r8. For example, if the instruction was swi 0x203, then r8 should contain 0x203.

c) Given the table of addresses shown below, write a sequence of instructions that will transfer control to the appropriate routine based on the contents of r8.

```
                .data
swi_functions:
                .word       =swi_00_routine
                .word       =swi_01_routine
                    ...
                .word       =swi_N_routine
```

Question 7 – 4 marks

Consider a hypothetical virtual memory system with eight (8) frames, with each frame holding 128 bytes. Each process has a virtual memory size of 2048 bytes.

a) How many bits are used to represent physical addresses in this system? How many bits are used to represent virtual addresses in this system?

b) Given the following subset of the page table, what is the physical address associated with the virtual addresses 0x1A3. If the the virtual address is invalid, write "invalid".

V	D	Frame
1	0	3
1	0	2
1	1	5
1	0	0
0	0	0
1	1	7

Question 8 – 4 marks

When allocating a two-dimensional array in C, it is common for compilers to lay out the memory in row-order. That is, given the following declaration:

```
int  a[1024][1024];
```

If `a[0][0]` is at memory location `0x100000`, then `a[0][1]` is at memory location `0x100004`, `a[0][2]` is at memory location `0x100008` and so on.

Considering the effects of cache and virtual memory, provide a **detailed explanation** why the program on the left is normally going to execute faster than the program on the right. Assume a page size of 4096 bytes.

`int  a[1024][1024];`
`for (int i=0;i<1024;i++)`
`{`
`    for (int j=0;j<1024;j++)`
`    {`
`      a[i][j] = i+j;`
`    }`
`}`
	`int  a[1024][1024];`
	`for (int i=0;i<1024;i++)`
	`{`
	`    for (int j=0;j<1024;j++)`
	`    {`
	`      a[j][i] = i+j;`
	`    }`
	`}`

Question 9 – 4 marks

Consider a hypothetical architecture where each instruction is broken down into three stages: fetch, decode and execute. In this architecture, each phase of instruction execution takes one clock cycle.

a) With a simple example, illustrate how instruction pipelining can increase instruction throughput compared to a non-pipelined architecture.

b) Explain how branch instructions complicate the pipelining process. Discuss how the ARM's conditional execution simplifies pipelining.

– END –