UNIVERSITY OF VICTORIA

EXAMINATIONS SPRING 2009

C SC 230 A01 – Introduction to Computer Architecture

NAME: _____ **STUDENT NO.:** _____

INSTRUCTOR: R. N. Horspool **DURATION:** 3 hours

TO BE ANSWERED ON THE PAPER

Question No.	Value	Mark	Question No.	Value	Mark
1	12		7	14	
2	15		8	14	
3	12		9	12	
4	16		10	12	
5	13		11	10	
6	10				
TOTAL MARKS				**140**	

INSTRUCTIONS:

1. STUDENTS MUST COUNT THE NUMBER OF PAGES IN THIS EXAMINATION PAPER BEFORE BEGINNING TO WRITE, AND REPORT ANY DISCREPANCY IMMEDIATELY TO THE INVIGILATOR
2. THIS EXAM HAS 14 PAGES, AND A ONE PAGE / 2-SIDED SEPARATE HANDOUT.
3. No aids are permitted. However, a 2-sided ARM instruction set reference page is provided for your use.
4. The marks assigned to each question are shown in square brackets. Partial marks are available for all questions.
5. Please be precise but brief, and use point form where appropriate.
6. It is strongly recommended that you read the entire exam through from beginning to end before beginning to answer the questions.

Question 1. [12] Fill in your answers in the right-hand column of the table below:

a)	What is the *binary* representation of the decimal number 623 ?	
b)	What is the *hexadecimal* representation of the decimal number 623 ?	
c)	What is the 8-bit two's complement *binary* representation of the decimal number -11 ?	
d)	What is the 32-bit two's complement *hexadecimal* representation of the decimal number -11 ?	
e)	What is the *decimal* equivalent of the hexadecimal number `0x20B` ?	
f)	A machine has 85 opcodes; how many bits are needed for the opcode field of an instruction on that machine ?	
g)	What is $(2^{14} \times 2^{13}) / 8$? (Give your answer as a power of two.)	
h)	An array A of 2-byte integers begins in memory at address `0x2ba8`; what is the address of the element A[15]? (Note: 15 is a decimal number, and the first array element is A[0].)	
i)	If R1 = `0x3E04` and R2 = `0x0E`, what memory location is accessed by the ARM instruction `LDR R0,[R1,R2, lsl #2]` ?	
j)	If R1 = `0x0007` initially, what new value is stored in R1 by the ARM instruction `ADD R1, R1, R1, lsl #2` ?	
k)	If R1 = `0x0007` initially, what new value is stored in R1 by the ARM instruction `RSB R1, R1, R1, lsl #3` ?	
l)	Simplify the following expression, giving your answer in hexadecimal: `(0x0A3D2 & 0xFF00F) | 0x3D0`	

Question 2. [15] Find the word or phrase from the list on the left that best matches the descriptions in the table on the right. Use the numbers to the left of words in the answer, and write them in the right-hand column of the second table. Each answer should be used only once. *Fill in the column labelled "#" below:*

1	status register
2	word
3	direct mapped cache
4	flash memory
5	program counter
6	MMU (memory management unit)
7	fully associative cache
8	an interrupt
9	decoder
10	opcode
11	link register
12	pipeline
13	DRAM (dynamic random access memory)
14	virtual memory
15	cycle stealing

	Description	#
A	Memory which retains its contents without power for an indefinite period.	
B	A register which holds the condition code	
C	A group of bits with the size used as data operands by most instructions.	
D	Component of the CPU where instructions are executed as a series of stages.	
E	A device which translates virtual addresses to real addresses.	
F	A logic circuit which generates a 1 value on one of its output lines as selected by the binary number provided as its input.	
G	A register which holds the return address from a subroutine	
H	A combination of RAM memory and disk memory which the program can access as though it were all RAM memory	
I	An asynchronous transfer of control to a system software routine.	
J	Memory whose contents must be repeatedly refreshed.	
K	A cache where any slot can hold any item.	
L	Means by which DMA (direct memory access) obtains its access to main memory	
M	A register which gives the address of the next instruction to execute	
N	A command to the CPU to perform an action.	
O	A cache where each item has a unique slot	

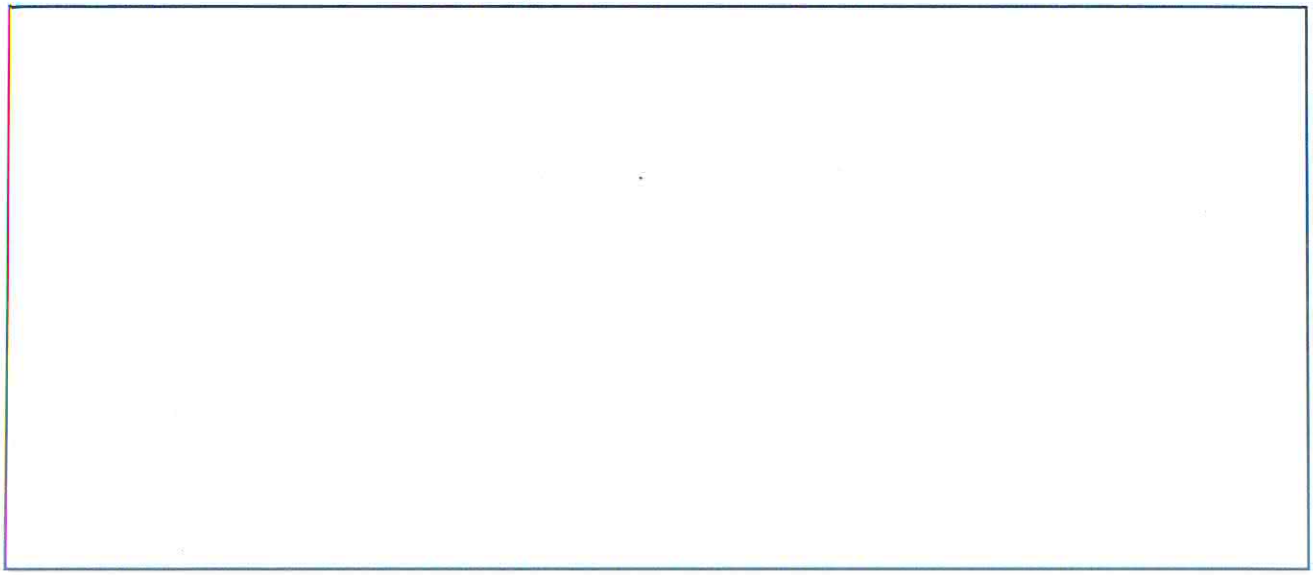
Question 3. [12] Figure 1 shows the main components of a CPU (which is *not* the ARM processor), its interface to memory via buses and its internal datapath. In the "Fetch/Decode/Execute" phases of an instruction, the "Fetch" phase is the same for every instruction. (The Fetch phase includes the operation of incrementing PC by 4; all instructions are 4 bytes in size.) The sequence of steps for the Fetch phase is shown in Table 1. Imitate the style and level of detail when answering parts (a) and (b) below.

a) **[6]** Give the detailed steps performed during the phases which follow Fetch for the instruction:

```
LDIND  R1, [R2]
```

where `LDIND` is the Load-Indirect instruction. In this example, register R2 holds the address of a word of memory, and that word holds the address of another word of memory. It is the word at that second address which is copied into register R1. The semantics can be expressed as:

```
R1 = Memory[ Memory[R2] ]
```



b) **[6]** Give the detailed steps performed during the phases which follow Fetch for the instruction:

```
BRX  R1
```

where `BRX` is the Branch-Relative-Indexed instruction. In the example, it causes an unconditional transfer of control to the intruction at the address computed as PC+R1.

Figure 1: A sample processor and its internal components

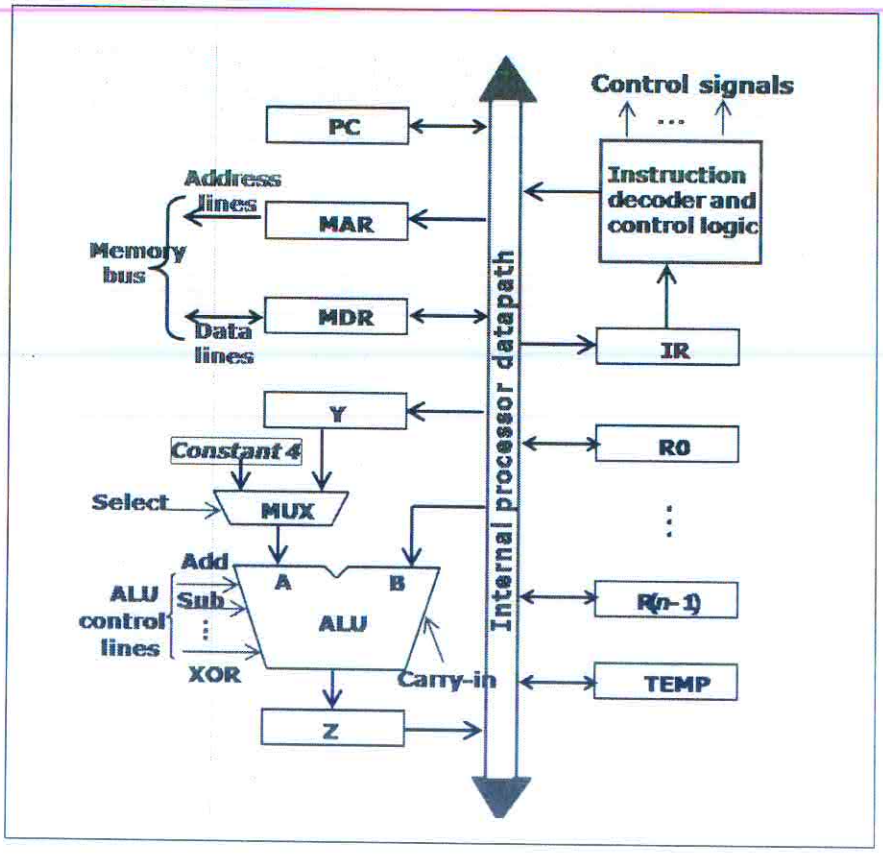


Table 1: Steps Performed during Fetch Phase of an Instruction

Step	Action
1.	PC → MAR
2.	MAR→ address bus
3.	Read signal → control lines
4.	PC → B input of ALU
5.	0 → MUX to select 4 as input

Step	Action
6.	Add signal → ALU control lines
7.	Z → PC
8.	Wait for memory to finish reading
9.	Data bus → MDR
10.	MDR → IR

Question 4. [16] You are to write ARM code for a function which checks whether parentheses are properly balanced inside a text string. They are balanced if the number of right parentheses never exceeds the number of left parentheses as the string is read from left to right and, at the end, the two numbers are equal. Some C code for this function is provided below. The function returns 1 if the parentheses are balanced and 0 otherwise.

```
int IsBalanced( char* str ) {
    int ix = 0; int count = 0;
    while( count >= 0 ) {
        char c = str[ix++];      /* get next character */
        if (c == 0)  /* have we reached the terminating byte? */
            if (count == 0) return 1 else return 0;
        if (c == '(') count++;
        if (c == ')') count--;
    }
    return 0; /* count is negative */
}
```

Write this function in ARM assembler. Do not provide code for the main calling routine. Assume that the string address (the `str` parameter) is passed in register R0. The result is to be returned in register R0. Your function is not allowed to have any side-effects. Make sure you include appropriate comments to receive full marks.

```
IsBalanced:      @ your ARM code begins here (continue on next page if needed)
```

```
@ (continuation of IsBalanced function)
```

Question 5. **[13]** Assume you have a system which contains various elements which have been discussed in the course, namely Virtual Memory, Cache, Main Memory, Pipelining, MMU (Memory Management Unit), TLB (Translation Lookaside Buffer), Page Table, DMA (Direct Memory Access). Answer the following questions:

a) **[1]** Where is the Page Table stored ?

b) **[1]** Where is the TLB stored ?

c) [3] Describe the steps which take place when there is a cache hit (the best case scenario).

d) [3] Describe the steps which take place when the needed data is still on disk (the worst case scenario).

e) [3] Describe the steps which take place when the needed data is in main memory, but not in the cache.

f) [2] You may or may not have mentioned the DMA in the steps above. If you included the DMA, give a definition of what it is and what function it had in the previous 3 scenarios. If you did not include the DMA, state why not and still give a definition of what it is.

Question 6. [10] This question is concerned with pipelining of instruction execution.

a **[2]** Give a one sentence explanation of instruction hazard.

b **[2]** Give a one sentence explanation of data hazard.

c **[2]** Give a one sentence explanation of structural hazard.

d) **[4]** Consider the sequence of ARM instructions listed below. Where in this sequence might pipelining hazard(s) occur – give the line number of the instruction, and state which kind of hazard it is. Assume that the branch in line 4 is taken.

```
1                ...
2                MUL     r1,r2,r3
3                SUBS    r1,r1,#1
4                BEQ     L1          @ taken!!!
5                STR     r1,[r4]
6       L1:      LDR     r5,[r5]
7                ...
```

Question 7. [14] Consider a cache memory with the following characteristics. The capacity of the cache memory is 2 kilobytes (this is storage for the cache lines only, the storage for tags and valid bits and dirty bits is extra). The line size is 32 bytes. The cache is 4-way set associative. It implements a write-back policy for its lines. Addresses on the computer are 24 bits in size, and memory is byte addressable.

a) **[1]** How many lines does the cache contain?

b) **[1]** How many sets does the cache contain?

c) **[2]** If the computer has 2^{24} bytes of RAM memory in total, how many blocks of RAM memory share the same set in the cache?

d) **[6]** Consider the memory address below, given in binary. Circle and label each of the following three components of the address: the **offset** within the line, the **set number**, and the **tag** (which will be saved in the cache alongside the cache line).

0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 0 0 0 1 1 1 1 0 0

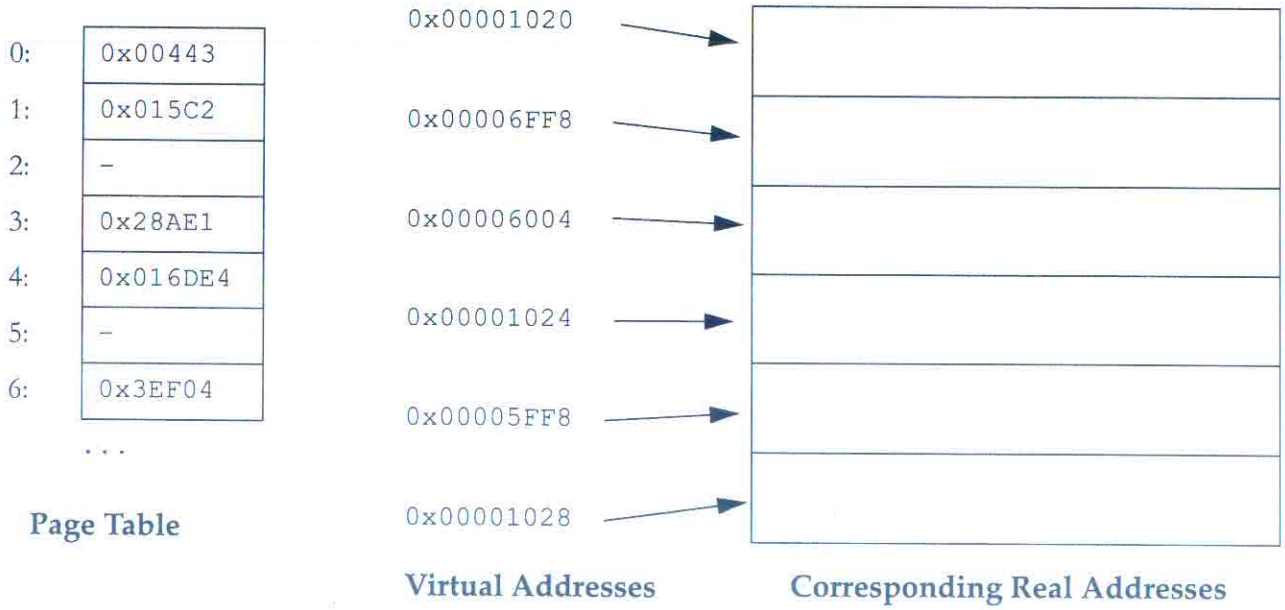
e) **[2]** When is the V (valid) bit set? When is it cleared?

f) **[2]** When is the D (dirty) bit set? How does this help implement a write-back policy?

Question 8. [14] Consider a computer which has a virtual memory system with the following characteristics. The computer has a 32-bit address space with byte addressable memory; this implies a virtual address space which is 2^2 bytes or 4 GB in size. The main memory (RAM memory) is 1 GB in size. Pages are 4 KB in size.

a **[2]** How many entries does the page table have?

b **[6]** Suppose that the CPU generates the sequence of memory addresses shown below (written in hexadecimal) and that the page table contains the entries shown on the left. Invalid (unmapped) entries in the page table are shown as empty. Show the corresponding sequence of addresses in main memory which the virtual addresses are mapped to. If any address cannot be mapped, show your answer as the words 'page fault'.



c) **[6]** Suppose that a program is running and incurs a page fault. All frames in the main memory are currently in use. What does the operating system have to do to allow the program to continue execution? Your answer should be written as a sequence of steps taken by the operating system.

Question 9. [12] Computer programs usually make use of library functions (or library subroutines). These functions may be either statically linked or dynamically linked.

a) [1] What software tool normally performs static linking?

b) [2] Give two advantages of static linking? over dynamic linking.

c) [2] Give two advantages of dynamic linking? over static linking.

d) [4] Dynamic linking requires the program to be built with stub routines instead of the actual functions or subroutines. What happens when the program first invokes one of these subroutines? List each of the steps.

e) [3] What happens when the stub routine is invoked a second time?

Question 10. [12] A very short mystery ARM assembler program is shown below. Label `Table` in the program is at address `0x1058` and `Result` is at address `0x1078`. Read the code and then answer the questions written alongside and below.

```
1                .text
2       _start:  ldr   r0, =Result
3                ldr   r1, =0x99A0000
4                bl    QQ
5                swi   0x11
6       QQ:      stmfd sp!,{r0-r5,lr}
7                mov   r3, #0
8                mov   r4, #0
9                ldr   r2, =Table
10      L1:      cmp   r1, #0
11               beq   L3
12               bgt   L2
13               ldrb  r5, [r2,r3]
14               strb  r5, [r0,r4]
15               add   r4, r4, #1
16      L2:      add   r3, r3, #1
17               mov   r1, r1, lsl #1
18               bal   L1
19      L3:      strb  r1, [r0,r4]
20               ldmfd sp!,{r0-r5,pc}
21               .data
22      Table:   .ascii
          "abcdefghijklmnopqrstuvwxyz012345"
23      Result:  .space  33
24               .end
```

a) [2] What value does r1 contain on the first occasion that line 18 is reached??

b) [1] What value does r1 contain on the second occasion that line 18 is reached?

c) [1] How many times does the `bal` instruction at line 18 jump to label L1?

d) [1] What value is stored in memory at line 19?

f) [4] The result of the function is an ASCII string in memory at label `Result`. What is that string?.

g) [2] How is the length of the ASCII string related to the function's input argument in r1? .

h) [1] Explain the number of bytes allocated for the `Result` array. Why is this the correct size?

Question 11. [10] You are given the following scenario:

- a processor has on-chip L1 cache and RAM memory accessible through the system bus;
- the execution time for a program is directly proportional to the memory access time;
- access to memory in the L1 cache takes 1 clock cycle, access to a location in the RAM memory takes 50 clock cycles;
- the hit rate for the L1 cache is 0.96;
- if a location is not found in the cache, it must first be fetched from main memory to the cache and then fetched from the cache to be executed – thus the total is 1 memory access + 1 cache access or 51 cycles.

a) **[6]** Compute the ratio of program execution time without the L1 cache to execution time with the L1 cache – this ratio is usually defined as the speed-up factor resulting from the presence of the cache.

b) **[4]** We now add a L2 cache to the computer's motherboard. The hit rate for the L1 cache remains at 0.96. However, when there is a L1 miss, the L2 cache is checked and it has a hit rate of 0.80. Access to a location in the L2 cache takes 5 clock cycles. If the memory location is found in the L2 cache, it takes the L1 cache access time + the L2 access time. If a location is not found in either cache, it will take the L1 access time + the L2 access time + the RAM memory access time. Now compute the ratio of program execution time without any cache to the execution time with both L1 and L2 caches.

END