CPEN 211 – Introduction to Microcomputers
Final Exam – December 2015

Time: 2.5 hours (150 minutes)

This examination consists of 16 pages. Please check that you have a complete copy. You may use both sides of each sheet if needed.

Surname First Name

Student Number

Signature

#	MAX	GRADE
1	23	
2	10	
3	5	
4	10	
5	10	
6	10	
7	10	
8	5	
TOTAL	83	

READ THIS

IMPORTANT NOTES:

1. By signing above you agree to uphold the highest standards of academic integrity during this exam. It is the policy of UBC APSC that any student suspected of cheating during an exam must have their booklet confiscated ***immediately***, before the end of the test, and be reported to the Dean's office.

2. The announcement "stop writing" will be made at the end of the examination. Writing after this announcement will be considered cheating and handled as above.

3. Instructions:

a) **Read each question carefully before attempting to solve it.**
b) **Attempt to solve <u>all</u> questions.**
c) **Keep all your answers on the exam sheet. Extra space is at the end and you may also write on the back of pages but you <u>must</u> add a note indicating you did this.**
d) **You are allowed two 8.5x11" single sided (or one double-sided), hand-written aid sheets aid sheet (not photocopied).**
e) **Absolutely NO communication or calculator devices allowed.**
f) **All exam booklet pages <u>must</u> be returned. You are not permitted to take any exam material from the exam hall. Removing pages is not recommended. If for any reason you do remove a page you MUST write your student number at the bottom or that page may not be marked.**
g) **There is a summary of some ARM syntax on the last page.**

Question 1 [23 marks]: Short answer. Each question is worth 1 mark unless specified.

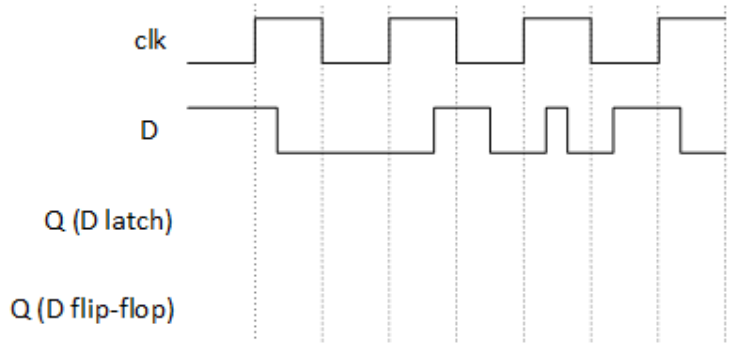
(a) What is the main advantage of using a digital logic circuit compared to an analog circuit when both can perform the same function?

(b) Find the dual function of the following function and write it in normal form: $f(x, y) = (x \wedge \bar{y}) \vee (\bar{x} \wedge y)$

(c) **[2 marks]** Draw a circuit for a *2-input* OR gate out of NFET and PFET transistors.

(d) List the *two* 1-bit signals connected to the input of the 2-input AND gate that drives output `b[18]` of a 6:64 decoder with output `b[63:0]` built out of one 2:4 decoder with output `x[3:0]` and one 4:16 decoder with output `y[15:0]`.

(e) **[2 marks]** Draw the output of a (level-sensitive) D latch and (positive edge-sensitive) D flip-flop given the following D and clk inputs:



(f) **[3 marks]** Draw a schematic diagram of read-write memory with 1-bit address storing 2-bit words showing as much detail as possible in the space below.

 Student number:_____

(g) What is 0xB1821433 interpreted as an ARM instruction?

(h) Encode **only** the BNE instruction in the ARM code below. Write it in hexadecimal.

```
Label:  LDR   R1,[R2,R0,LSL#4]
        STR   R1,[R3,R0,LSL#2]
        ADD   R0,R0,#1
        CMP   R0,R4
        BNE   Label
```

(i) Encode -4.5625 in IEEE single-precision floating-point. Write it in hexadecimal.

(j) **[2 marks]** What is the 4-bit sum of the 2's complement values 1101_2 and 1010_2 and does this sum generate an overflow (yes/no)?

(k) **[2 marks]** List **2** differences between an interrupt service routine and a function call.

(l) Briefly, summarize the purpose and operation of an interrupt controller such as the ARM Generic Interrupt Controller (GIC) used in Lab 10.

(m) What is the difference between an edge-sensitive and level-sensitive interrupt?

(n) List a sequence of <u>five</u> addresses resulting in a *larger* number of cache *misses* for a fully-associative (FA) cache with two 1-byte blocks and LRU replacement than for a direct-mapped (DM) cache with two 1-byte blocks (i.e., FA does *worse* than DM).

(o) How much would you expect the miss rate of a cache to decrease if the size of the cache is doubled?

(p) **[2 marks]** What fraction of a program must be able to run in parallel across all 100 cores of a 100-core processor to achieve a speedup by a factor of 10?

Question 2 [10 marks]: Draw a logic diagram equivalent to the following Verilog using only D-flip-flops, AND, OR, NOT, XOR gates and/or multiplexers. To obtain full marks highlight any multiplexers implicit in the circuit on your logic diagram.

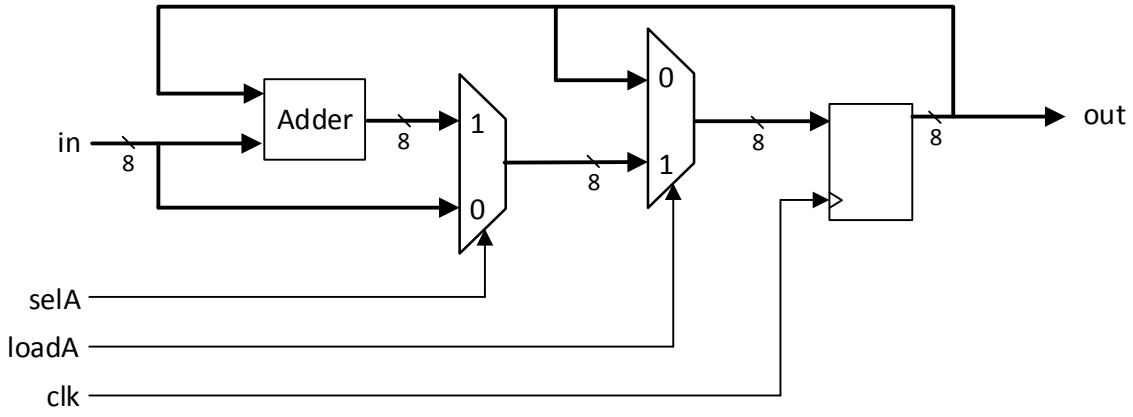
```
module q2(a,b,c,d);
  input a, c;
  input [1:0] b;
  output d;
  wire [3:0] e, f;
  reg [3:0] g;
  r #(2,4) U(b,e);
  assign d = |(e & f);
  always @(posedge c)
    g = f;
  wire [2:0] s = {e[3], e[2:1] | s[2:1]};
  assign f = {(s & g[2:0]) | (~s & g[3:1]), a};
endmodule

module r(a,b);
  parameter n=1;
  parameter m=2;
  input [n-1:0] a;
  output [m-1:0] b;
  assign b = 1 << a;
endmodule
```

There is additional space for this question on the next page

(extra space for Question 2)

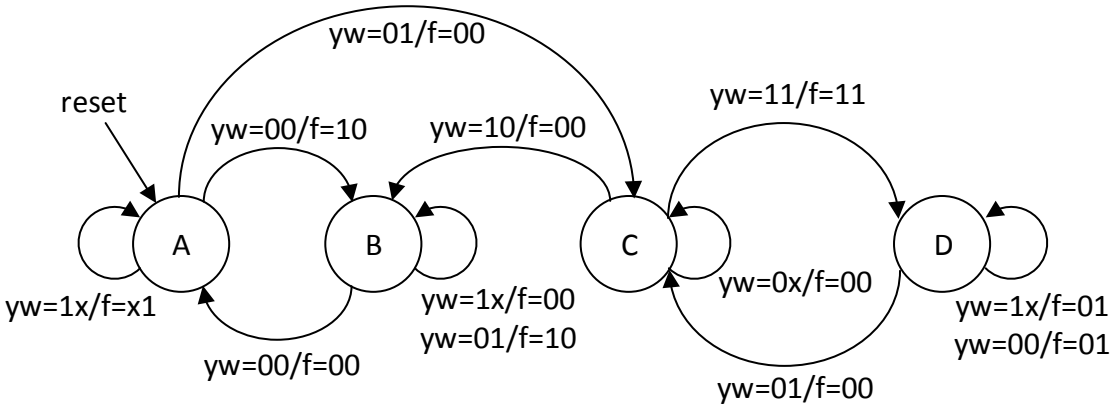
Question 3 [5 marks]: Write synthesizable Verilog equivalent to the following circuit. You may use modules defined in the textbook, but for full marks you must write out their definition. Your Verilog must follow the CPEN 211 style guidelines. Overly complex solutions may have marks deducted even if correct.




```
module q3(in,selA,loadA,clk,out);
```

 Student number:_____

Question 4 (a) [5 marks]: Write Verilog to implement the following Mealy state machine with inputs **y** and **w** and 2-bit output **f**. Even though we did not do a Mealy example in class you should be able to do this question. As output **f** depends *both* on the current state and the input it is specified on the edges. The reset state is A. Your Verilog must follow the CPEN 211 style guidelines. Assume vDFF is already defined as in class.




```
module q4(clk,reset,y,w,f);
```

 Student number:_____

Question 4 (b) [5 marks]: Create a digital circuit that implements the FSM from Question 4 (a). Your final solution should be in the form of a logic diagram containing only AND, OR, NOT gates and flip-flops. You do **not** have to include reset logic.

Question 5 [10 marks]: Write ARM assembly for the C code below following the ARM calling convention. This code can be used to compute DNA sequence similarity.

```
int cost( char *s1, unsigned n1, char *s2, unsigned n2 )
{
   if( n1==0 ) { return 2*n2; }
   if( n2==0 ) { return 2*n1; }
   int cm = cost(s1+1, n1-1, s2+1, n2-1);
   if( *s1 != *s2 ) { cm = cm + 1; }
   int c1 = cost(s1+1, n1-1, s2,   n2)   + 2;
   int c2 = cost(s1,   n1,   s2+1, n2-1) + 2;
   int result = (c1 < c2) ? c1 : c2;
   result = (result < cm) ? result : cm;
   return result;
}
```

Additional Notes: The C statement "`d = a ? b : c`" assigns `d` the value of `b` if `a` is not zero and `c` otherwise. A `char` is 8-bits. The notation "`char *x`" means that `x` is a variable of type "pointer to `char`". The value of a "pointer to char" represents a memory address of a location containing a char value. If `x` is a variable declared as a pointer the notation "`*x`" as used in the code above means, "read the value in memory at the address `x`". Adding an integer to a pointer value results in a pointer value.

There is additional space for this question on the next page

(extra space for Question 5)

Question 6 [10 marks] Convert the following C code fragment into ARM assembly.

```
for( n=0; n < N; n++ ) {
   double force = 0.0;
   for( m=0; m < N; m++ ) {
     if ( m != n ) {
       double d = p[m][0] - p[n][0];
       double sign = 1.0;
       if( d < 0 ) { sign = -1.0; }
       force += sign * g * ( p[n][2] * p[m][2] ) / ( d * d );
     }
   }
   p[n][1] += force * dt / p[n][2];
}
```

Assumptions: Array "p" is declared as "`double p[N][3];`" and the base address of array "p" is initially in register R0. Register R1 contains the value N. `g` is in register D0. `dt` is in register D1. N is greater than zero. The variables `n` and `m` are 32-bit integers.

Hints: Recall that "`x += y`" means the same as "`x = x + y`". Use floating-point mnemonics (e.g., FLDD, etc…) for floating-point operations. Use FCMPD to compare two double precision floating-point registers.

There is additional space for this question on the next page

(extra space for Question 6)

Question 7 (a) [4 marks] Assuming a 20-bit address space, how many block offset bits, index bits, and tag bits are required for a 2-way set associative, 256B cache with 32B blocks?

Block offset bits = _____

Index bits = _____

Tag bits = _____

Minimum page size required to avoid the synonym problem = _____

Question 7 (b) [6 marks] Consider a system including both a cache and TLB. Consider a 20-bit virtual address space; 12-bit physical address space, one-level page table with 256B pages; 2-entry, fully associative TLB with LRU replacement policy; 256B 2-way set associative data cache with 32B blocks using a virtually-indexed and physically-tagged organization. The cache is write-back with write allocate and LRU replacement. Assume page table entries are not cached in the data cache.

The contents of the page table are illustrated below. On a page fault the next physical page allocated is 7, 8, 9, etc…

virtual page #	resident?	physical page #
0	*yes*	*6*
1	*no*	*-*
2	*no*	*-*
3	*no*	*-*

The table below contains a sequence of memory references and the cycle on which they occur. "Op" means whether the access was a read or write. Using the information above **fill out the missing information in the table below**. **State any assumptions.**

Cyc	Op	Virtual Address	Physical Address	TLB Hit ?	$ Hit ?	Write back?	# Mem Access	Page Fault ?
1	Write	0x00008						
2	Write	0x00100						
3	Read	0x00180						
4	Read	0x00008						
5	Write	0x00194						
6	Read	0x00320						
7	Read	0x00230						
8	Read	0x00008						

Question 8. Your company has an ARM license enabling it to develop its own CPU that implements the ARMv7 instruction set. The CPU hardware design is not yet complete. Management asks the software team you are on to estimate the execution time an important program. You have the following information:

CPI for various instructions on your company's ARMv7 implementation:

Instruction Type	**CPI**
ALU (ADD, SUB, etc…)	1
Branches	3
Memory (Load, Store)	5
Floating-Point	10

Question 8 (a) [2 marks] Suppose that the important program is composed of 10% floating-point instructions, 30% memory operations and 15% branches and the rest ALU operations. What is the average CPI?

Question 8 (b) [2 marks] If the important program consists of 1 million instructions and must execute in 0.002 seconds or less then what is the minimum clock frequency that the hardware team needs to achieve in their design of your company's new CPU design? Assume the same instruction frequencies as in Part (a).

Question 8 (c) [1 marks] Explain how "forwarding" can improve performance.

(extra space – clearly indicate question number)

ARM Summary

Data Processing (op): ADD (4), SUB (2), AND (0), ORR (12), MVN (15), MOV (13)
Data Transfer (op): LDR (25), STR (24), LDR *shifted Rm* (57), STR *shifted Rm* (56)
Registers: R0-R15: stack pointer is R13, link register is R14, PC is R15; CPSR (status: bits 31-28 are N, Z, C, V flags)
Shifted Operands (shift): LSL (00), LSR (01)

***Instruction Encodings**:*

Instruction	Format	Cond (4 bits)	F (2)	I (1)	Op (4/6)	S (1)	Rn (4)	Rd (4)	Operand2 (12)
ADD, etc…	DP	cond	0	imm?	op	0	reg	reg	imm / reg, shift
LDR, etc…	DT	cond	1	n.a.	op	n.a.	reg	reg	imm / reg, shift
			Op (4)		**Target (24)**				
B<suffix>		cond	10		$=(PC_{target}-(PC_{branch}+8))/4$				
			Op (8)				**Rn (4)**	**(4)**	**Operand2 (12)**
CMP		cond	0x15				reg	0	reg
CMP (imm)		cond	0x35				reg	0	imm

***Shift Encoding** (Operand2 in instruction encoding)*

11	*10*	*9*	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
shift_imm					*shift*		*0*	*Rm*			
Rs				*0*	*shift*		*1*	*Rm*			

***Immediate Encoding**: value*=bits 0:7 as unsigned (0-255). If *r*=bits 8:11≠0 then also rotate this *value* right by 2×r. "Rotate right" means the least significant bit is copied to the most significant bit. E.g., 0001 rotated right by 1 is 1000 and rotated right by 2 is 0100.

***Condition codes** (use to determine value for "cond" field in instruction encoding):*

Suffix	Flags	Meaning	cond
EQ	Z	Equal	0
NE	~Z	Not Equal	1
HS	C	Unsigned >=	2
LO	~C	Unsigned <	3
MI	N	Negative	4
PL	~N	Positive	5
VS	V	Overflow	6
VC	~V	No overflow	7

Suffix	Flags	Meaning	cond
HI	C&~Z	Unsigned >	8
LS	~C | Z	Unsigned <=	9
GE	N = V	Signed >=	10
LT	N != V	Signed <	11
GT	~Z&N=V	Signed >	12
LE	Z | N!=V	Signed <=	13
AL	-	Always	14
NV	-	Reserved	15

ARM calling convention:

Name	Registers	Usage	Preserved on call?
a1-a2	R0-R1	Argument / return result / scratch register	no
a3-a4	R2-R3	Argument / scratch register	no
v1-v8	R4-R11	Variables for local routine	yes
ip	R12	Intra-procedure-call scratch register	no
sp	R13	Stack pointer	yes
lr	R14	Link Register (Return address)	yes
pc	R15	Program Counter	n/a