McGill University
School of Computer Science

Introduction to Computer Systems

Midterm Examination - **Answers**

October 25, 2012 – In Class
Instructor: Joseph Vybihal

Student Name: _____ Student ID: _____

Instructions

- No notebooks, calculators or textbooks permitted in this exam.
- Language translation dictionaries are permitted.
- You are permitted to write your answers in either English or French.
- Attempt all questions.
- **Parts marks** are given for all questions - **show your work**.
- All answers must be written on the examination sheets

Grading

Section	Grade	Your Mark
Question 1: Memory and Number Systems	20	
Question 2: Circuit Problem	40	
Question 3: Micro-instructions	20	
Question 4: Computational Problem	20	
Total	100 %	

YOU MAY WRITE YOUR ANSWERS ON BOTH SIDES OF THE EXAM SHEETS

YOU MUST HAND IN YOUR EXAM AT THE END OF THE EXAMINATION

SUPPLEMENTARY TABLES PROVIDED FOR YOU AT THE END OF THE EXAM

Question 1: Computer Memory and Number Systems

Do the following:

1. Assume we have a segment of RAM defined for us starting from address 100 until, and including, address 120. Assume this machine's addresses are 16-bits long. Draw a block diagram of this memory (NOT a circuit diagram). Make it large enough so that you can write within the cells. Specify the width in bits of various parts of your block diagram. Only provide a diagram for the segment of RAM specified, ignore all other elements.
2. What would the C string “Hi Bob” look like if it were placed at address 110.
3. What would the integer number -2 (negative two) look like if placed at address 100.

ADDRESS – 16 Bits	DATA – 8 Bits
120	
119	
118	
117	
116	'00000000
115	'01100010
114	'01101111
113	'01000010
112	'00100000
111	'01101001
110	'01001000
109	
108	
107	
106	
105	
104	
103	
102	
101	'11111110
100	'11111111

Question 2: Circuit Problem

You may use any of the black boxed "off-the-shelf" circuits we covered in class. Make sure to use their correct diagrammatic representation. You, of course, can use the standard AND, OR and NOT gates. Make sure to use their correct diagrammatic representation.

Assume we have a quartz crystal that ticks at exactly 1 tick per second. At each tick is emits an electrical pulse, which we will interpret as a 1. This is emitted for a quarter of a second and then reverts to 0 for the remainder of the second. Assume three are three registers given to you. The first register is 5 bits long, while the other two registers are 6 bits long. These three registers store the time when the alarm must sound. The 5-bit register stores the hour in 24-hour time. The other two registers are the minutes and seconds (from 0 to 59). Assume that there exists some mechanism to set these registers to the time the user would like the alarm to sound, but we will not provide any circuitry for this. Assume, finally, we have an alarm that takes as input an input wire that when it receives a 1 will sound an alarm for 60 seconds and then stops.

Create a circuit that will sound the alarm at the time provided for 60 seconds. You may assume that your clock will begin counting from 0 hours 0 minutes and 0 seconds.

I will not provide a solution for this one because there are many valid solutions. If you think yours runs properly or deserves ore marks then please come and see me during my office hours.

Question 3: CPU Micro-instructions (40 points)

Provide the (1) classical and (2) pipeline micro-instructions for the command:

LoadWord RegisterNumber, RAMaddress

CLASSICAL

IR ← RAM[PC] & PC ← PC + 1 (or: MAR ← PC then MBR ← RAM[MAR], IR ← MBR)
CU ← IR(opcode)
MAR ← IR(address)
MBR ← RAM[MAR]
Rx ← MBR

PIPELINE

IR ← InstructionCache[PC] & PC ← PC + 4
CU ← IR(opcode)
MAR ← IR(address)
MBR ← RAM[MAR]
L ← MBR & R ← 0
A-OUT ← ALU(L,R)
Rx ← A-OUT

Question 4: Computational Problem (40 points)

Assume we have a peripheral that wants to send integers to the CPU to be summed immediately upon reception of the number. The sum will be stored in one of the CPU's general purpose registers. Assume the program is performing the following instructions:

```
LoadWord  Register1, PERIPHERALaddress
ADD       Register2, Register2, Register1    // Register2 is the sum
JUMP      To the LoadWord instruction above
```

What is the maximum speed the peripheral can transmit it's numbers? The user does not want to loose any values, since the transmission speed can be adjust from the peripheral.

Assume we have a classical CPU that is shaped like a pipeline CPU (Fetch, Load, ALU, Store). In other words, we have a pipeline CPU that processes on 1 instruction at a time.

Assume the following: System Board bus speed 20 ns per byte. Fetch 1 ns. Load register 1 ns. Load from memory depends on System Board. ALU operation 4 ns. Store cache or register operation 2 ns.

Assume all other time delays are negligible.

Answer:

- **Need to compute the length, worst case, for the entire loop.**
- **Note: a pipeline CPU with all its stages, BUT only 1 instruction passes through completely before the next instructions is executed.**

- **LW = fetch, load from RAM, ALU step, store to register**
- **ADD = fetch, load from register, ALU step, store to register**
- **JMP = fetch, load PC**

- **fetch = 1 ns, LoadReg = 1 ns (parallel), Load RAM = 20 ns per byte (4 byte word!), ALU = 4 ns, and store register = 2 ns**

- **lw = 1 + (20 * 4) + 4 + 2 = 87 ns**
- **add = 1 + 1 + 4 + 2 = 8 ns**
- **jmp = 1 + 1 = 2 ns**

- **Total = 97 ns**

SUPPLEMENTARY TABLES

ASCII Code: Character to Binary

0	0011 0000	O	0100 1111	m	0110 1101
1	0011 0001	P	0101 0000	n	0110 1110
2	0011 0010	Q	0101 0001	o	0110 1111
3	0011 0011	R	0101 0010	p	0111 0000
4	0011 0100	S	0101 0011	q	0111 0001
5	0011 0101	T	0101 0100	r	0111 0010
6	0011 0110	U	0101 0101	s	0111 0011
7	0011 0111	V	0101 0110	t	0111 0100
8	0011 1000	W	0101 0111	u	0111 0101
9	0011 1001	X	0101 1000	v	0111 0110
A	0100 0001	Y	0101 1001	w	0111 0111
B	0100 0010	Z	0101 1010	x	0111 1000
C	0100 0011	a	0110 0001	y	0111 1001
D	0100 0100	b	0110 0010	z	0111 1010
E	0100 0101	c	0110 0011	.	0010 1110
F	0100 0110	d	0110 0100	,	0010 0111
G	0100 0111	e	0110 0101	:	0011 1010
H	0100 1000	f	0110 0110	;	0011 1011
I	0100 1001	g	0110 0111	?	0011 1111
J	0100 1010	h	0110 1000	!	0010 0001
K	0100 1011	I	0110 1001	'	0010 1100
L	0100 1100	j	0110 1010	"	0010 0010
M	0100 1101	k	0110 1011	(	0010 1000
N	0100 1110	l	0110 1100	)	0010 1001
				space	0010 0000