

Name: _____ Email: _____

Problem 1: A group of EECS students have decided to compete with Motorola Corporation in the embedded DSP wireless network market. The RISCEE3 computer is a **16 bit** single-cycle computer.

There is only **1 register** (i.e. accumulator, called A). The PC and alu are eight bits wide.

Note: You can remove and keep the RISCEE3 and RISCEE4 diagrams from the exam.

Remember: anything AND with zero is always zero. Anything OR with one is one.

There is only one instruction format shown as follows:

Opcode	Data8 or Address8 field
8 bits	8 bits
15-8	7 - 0

(a) (12%) Fill in the settings of the control lines determined by the all the instructions **(use X for Don't Care)**

Machine Instruction	Operation	RegDst	ALU	MemWrite	RegWrite	BZ	P0
clear	A = 0	2	4	0	1	0	0
addi data8	A = A + data8	2	5	0	1	0	0
add addr8	A = A + Memory[addr8]	0	3	0	1	0	0
store addr8	Memory[addr8] = A	X	3	1	0	0	0
bne addr8	If (A != 0) { PC = addr8;}	X	1	0	0	1	0
apc	A = PC+2	3	X	0	1	0	0

(b) (20%) Using the above instruction set, fill in the code for the pseudo-instructions

Pseudo-Instruction	Operation	Assembly machine instructions (from part 1a)
loadi data8	A = data8	clear # A=0; addi data8 # A=0+data8
load address8	A = Memory[addr8]	clear # A=0; add addr8 # A=0+Memory[addr8]
jmp address8	pc = address8	clear # A=0; addi 1 # A=1; always not zero bne addr8 # if (1 != 0) { pc = addr8; } #alternate solution: apc #A=PC; Assume PC not zero bne addr8 #always branch
jal address8	A=PC+x; PC=address8 <i>Where A really contains the return address after the pseudo instruction returns. (Assume PC never becomes zero)</i>	apc #A = PC+2: addi 4 #...+ length of addi and bne bne addr8 #←return address here (=A)!

Problem 2: The Credit Suisse First Boston investment bank will only invest in the multi-cycle RISCEE4 architecture for the machine instructions of problem 1a. **Use X for Don't Care**. Assume parts 2a, 2b, 2c **are independent of each other**. Assume the 8 bit memory system is smart and loads the proper 16 bits in the IR register in one memory read cycle.

(a) (10%) Fill in the settings of the control lines needed for the “clear” instruction.

Clock Step	Mem Write	Mem Read	lorD	IR write	P0	BZ	PC src	ALU op	ALU srcA	ALU srcB	Reg Write	Reg Dst	
T ₁	0	1	1	1	1	X	1	5	0	0	0	X	fetch
T ₂	0	0	X	0	0	0	X	X	X	X	0	X	decode
T ₃	0	0	X	0	0	0	X	4	X	X	X	X	Aluout=0
T ₄	0	0	X	0	0	0	X	X	X	X	1	0	A=aluout

T₁ and T₂: RegWrite could equal X here because clear instruction will overwrite it later in T₄ but since are used by all other instructions also at T₁ and T₂, RegWrite must equal 0.

Alternate solution: Merge T₂ & T₃ from above. During instruction decode set ALUOut to zero

Clock Step	Mem Write	Mem Read	lorD	IR write	P0	BZ	PC src	ALU op	ALU srcA	ALU srcB	Reg Write	Reg Dst	
T ₁	0	1	1	1	1	X	1	5	0	0	0	X	fetch
T ₂	0	0	X	0	0	0	X	4	X	X	0	X	Decode Aluout=0
T ₃	0	0	X	0	0	0	X	X	X	X	1	0	A=aluout

(b) (10%) Fill in the settings of the control lines needed for “add” from memory instruction.

Clock Step	Mem Write	Mem Read	lorD	IR write	P0	BZ	PC src	ALU op	ALU srcA	ALU srcB	Reg Write	Reg Dst	
T ₁	0	1	1	1	1	X	1	5	0	0	0	X	fetch
T ₂	0	0	X	0	0	0	X	X	X	X	0	X	decode
T ₃	0	0	X	0	0	0	X	1	X	3	0	X	Aluout= IR[7:0]
T ₄	0	1	0	0	0	0	X	X	X	X	0	X	Memread
T ₅	0	0	X	0	0	0	X	5	1	1	0	X	ALUout= mdr+a
T ₆	0	0	X	0	0	0	X	X	X	X	1	0	A=aluout

Alternate solution: Merge T₂ & T₃ from above. During instruction decode set ALUOut to addr8

Clock Step	Mem Write	Mem Read	lorD	IR write	P0	BZ	PC src	ALU op	ALU srcA	ALU srcB	Reg Write	Reg Dst	
T ₁	0	1	1	1	1	X	1	5	0	0	0	X	fetch
T ₂	0	0	X	0	0	0	X	1	X	3	0	X	Decode Aluout= IR[7:0]
T ₃	0	1	0	0	0	0	X	X	X	X	0	X	Memread
T ₄	0	0	X	0	0	0	X	5	1	1	0	X	ALUout= mdr+a
T ₅	0	0	X	0	0	0	X	X	X	X	1	0	A=aluout

(c) (10%) Fill in the settings of the control lines needed for “bne” instruction

Clock Step	Mem Write	Mem Read	lorD	IR write	P0	BZ	PC src	ALU op	ALU srcA	ALU srcB	Reg Write	Reg Dst	
T ₁	0	1	1	1	1	X	1	5	0	0	0	X	fetch
T ₂	0	0	X	0	0	0	X	X	X	X	0	X	decode
T ₃	0	0	X	0	0	1	2	1	1	X	0	X	PC= (A!=0)? IR[7:0];

(d) (12%) Fill in the critical path times for each instruction. The delay time of the functional units are as follows Memory Write 8 ns, Memory Read 5 ns, Register (read or write) and opcode decode 1 ns, and ALU & Adders 2 ns.

Instruction	Instruction memory	Decode & Register Read	1 st ALU operation	Data Memory	2 nd ALU operation	Register Write	Total Time	Clock Cycles
clear	5	1	2			1	9	4
alternate clear	5	1				1	7	3
addi	5	1	2			1	9	4
add	5	1	2	5	2	1	16	6
store	5	1	2	8			16	4
bne	5	1	2*				8	3
apc	5	1	2			1	9	4

* Branch needs 2ns to compute zero value detect in ALU.

(e) (8%) Determine the fastest clock speed for the computer to work properly in frequency and show why.

{ Graders: use the slowest resource from part 2d in columns 2 to 7 }

Clock period is the slowest resource in any one step: 8 ns
Clock frequency = 1/period = 1/8ns = 125 Mhz

(f) (18%) Fill in the Clock, CPI, and MIPS in the above table and show all calculations.

{ Graders: use the student's own data from part 2d and 2e. Grade only CPI and MIPS }

Instruction	Clock Cycles	Instruction Mix
clear	4	10%
addi	4	30%
add	6	20%
store	4	10%
bne	3	5%
apc	4	25%
Clock speed	125 MHz	
CPI	4.35	
MIPS	28.7	

$$4 * (10\% + 30\% + 10\% + 25\%) + 3 * 5\% + 6 * 20\% = 4 * 75\% + 3 * 5\% + 6 * 20\%$$

$$125 \text{ MHz} / 4.35$$

Alternate solution: Clear = 3 clocks

$$\text{CPI} = 4.25 = 4 * (30\% + 10\% + 25\%) + 3 * (10\% + 5\%) + 6 * 20\%$$

$$\text{MIPS} = 29.4 = 125 \text{ MHz} / 4.25$$

The technology group of *Credit Suisse First Boston*, www.tech.csfb.com, would like you to have access to the following *extra credit* which can be only used for this exam and the previous exam.

{ Graders: Extra credit solutions are either right or wrong. No partial credit. }

a) (3%). Assemble the following machine instruction into binary & is located at address **0x17081812**

Field 1	Fields 2 and etc.	MIPS instruction
000011	00 0000 0000 0000 0000 0000 1101 = 0x17081868 - (0x17081812 + 4)>>2 = 0xd	jal 0x17081868

* Alternate (correct answer): the instruction is not located on a word boundary.

b) (2%) Give the two's complement of the **12 bit signed binary 0x911**

$$-0x911 = (-0x911+1) = (\sim 100100010001+1) = (011011101110+1) = \mathbf{011011101111} = \mathbf{0x6EF}$$

c) (2%) Convert **-17** into a **5 bit signed binary**. **Not possible, cannot fit in 5 bits**

$$17 = 16+1 = 10001; \sim 17 = 01110; \sim 17+1 = 01111; \text{incorrect sign bit}$$

d) (2%) Convert the **6 bit signed binary 111001** into decimal

$$1 \cdot (-32) + 1 \cdot 16 + 1 \cdot 8 + 0 \cdot 4 + 0 \cdot 2 + 0 \cdot 1 = -32+16+8+1 = \mathbf{-7}$$

e) (5%) Add **0xF** and **0xF** and what is the **signed overflow bit = 0** = Cout₃ ^ Cin₃

Cin	1	1	1	
	1	1	1	1
	1	1	1	1
Sum	1	1	1	0
Cout	1	1	1	1

f) (3%) Multiply the 2 bit unsigned binary numbers **11** by **11** into a 4 bit unsigned binary number.

$$\begin{array}{r}
 \\
 \\
 \\
 \hline
 1
 \end{array}$$

g) Assume the register size is **2-bits** and contain some symbolic values, so that \$s1=x and \$s2=y.

	Symbolic contents of \$s1	Symbolic contents of \$s2
	X =11	Y =10
add \$s1,\$s1,\$s2	X+Y =01	Y =10
sub \$s2,\$s1,\$s2	X+Y =01	X = (X+Y) - Y = 11
sub \$s1,\$s1,\$s2	Y = (X+Y) - X = 00	X =11

(a) (3%) Fill in the symbolic values of the registers.

(b) (3%) What does this code symbolically do? **Swap the contents of \$s1 with \$s2**

(c) (3%) Will the code work for these 2-bit binary values, when \$s1=11 and \$s2=10 and give reason. **After the swap \$s1 should be 10 and \$s2 should be 11 but because the overflow or the register size is too small, this symbolic swap give the wrong numerical results of \$s1=00 and \$s2=11**