EECS 314 Test 3 solutions Wednesday April 3, 2002

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) **(XX%)** Fill in the settings of the control lines determined by the all the instructions **(use X for Don't Care)**
{ See architecture solutions on powerpoint slides }

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) (XX%) 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=addess8 *Where A really contains the return address after the pseudo instruction returns.* ***(Assume PC never becomes zero)***	**apc #A = PC+2:** **addi 4 #…+ length of addi and bne** **bne addr8** **#←return address here (=A)!**

Problem 2: RISCEE4 is the multi-cycle 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.

{ See architecture solutions on powerpoint slides }

(a) (XX%) Fill in the settings of the control lines needed for the **"clear"** instruction.

Clock Step	Mem Write	Mem Read	IorD	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

T_1 and T_2: RegWrite could equal X here because clear instruction will overwrite it later in T_4 but since are used by all other instructions also at T_1 and T_2, RegWrite must equal 0.

Alternate solution: Merge T_2 & T_3 from above. During instruction decode set ALUOut to zero

Clock Step	Mem Write	Mem Read	IorD	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) (XX%) Fill in the settings of the control lines needed for **"add"** *from memory* instruction.

Clock Step	Mem Write	Mem Read	IorD	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 T_2 & T_3 from above. During instruction decode set ALUOut to addr8

Clock Step	Mem Write	Mem Read	IorD	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) (XX%) Fill in the settings of the control lines needed for **"bne"** instruction

Clock Step	Mem Write	Mem Read	IorD	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) (XX%) 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	1st ALU operation	Data Memory	2nd 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) (XX%) 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) (XX%) 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**	4*(10%+30%+10%+25%)+3*5%+6*20%=4*75%+3*5%+6*20%	
MIPS	**28.7**	125 MHz/4.35	

Alternate solution: Clear = 3 clocks

CPI = **4.25** = 4*(30%+10%+25%)+3*(10%+5%)+6*20%

MIPS = **29.4** = 125MHz/4.25

Problem 3. Assume a simple **6 stage** pipeline with the following execution times

1	IF	Instruction fetch 1st part	3 ns
2	IS	Instruction fetch 2nd part	4 ns
3	ID	Register Read	2 ns; **Branch decision made here**
4	EX	ALU	3 ns
5	MEM	Data Access	8 ns,
6	WB	Register Write	2 ns

This computer has the following instructions:

Instruction	Operation
add $rd, $rs, $rt	$rd = $rs + $rt
beq $rs, $rt, disp16	pc = pc+2+($rs-$rt=0?disp16:0)
lw $rt, addr16($rs)	$rt = Mem[addr16+$rs]
sw $rt, addr16($rs)	Mem[addr16+$rs]=$rt

3a (XX%) Fill in the following tables

Instruction	IF	IS	ID	EX	MEM	WB	Total Time	Multi-Cycles	Instruction Mix
add	**3**	**4**	**2**	**3**		**2**	**14**	**5**	50%
beq	**3**	**4**	**2**				**9**	**3**	25%
lw	**3**	**4**	**2**	**3**	**8**	**2**	**22**	**6**	17%
sw	**3**	**4**	**2**	**3**	**8**		**20**	**5**	8%

3b (XX%) Fill the following table and show work.

Instruction	Clock frequency	CPI (no hazards)	MIPS (no hazards)
Single-cycle CPU	1/22ns **=45.5 MHz**	**=1**	45.5/1 **= 45.5 MIPS**
Multi-cycle CPU	1/8ns **=125 MHz**	5*50% + 3*25% + 6*17% + 5*8% **=4.67**	125/4.67 **=26.8 MIPS**
Pipeline CPU	1/8ns **=125 MHz**	(no hazards = no dependancies = no stalls) **=1**	125/1 **=125 MIPS**

The following parts refer to the pipelined machine only
For the following code: Assume no forwarding.

3c (XX%) For the following code: Assume no forwarding and no branch prediction.
Draw lines showing all the data dependencies and show the pipeline sequence (IF,IS,ID,EX,M,WB)
Note: Branch decision is made in the ID stage.

Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
sub $3, $4, $5	**IF**	**IS**	**ID**	**EX**	**M**	**W**										
beq $1, $3, loop		**IF**	**IS**	**ID**	**ID**	**ID**	**EX**	**M**	**W**							
add $4, $4, $5			**IF**	**IS**	**IS**	**IS**	**ID**	**EX**	**M**	**W**						

3d (XX%) Draw lines showing all the data dependencies in "Time" column.
and show the 6-stage pipeline sequence (IF, IS, ID, EX, M, WB) for the following code

Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
sw $1, 4($2)	**IF**	**IS**	**ID**	**EX**	**M**	**W**										
sub **$4**, $1, $2		**IF**	**IS**	**ID**	**EX**	**M**	**W**									
add **$5**, **$4**, $2			**IF**	**IS**	**ID**	**ID**	**ID**	**EX**	**M**	**W**						
lw $3, 8(**$5**)				**IF**	**IS**	**IS**	**IS**	**ID**	**ID**	**ID**	**EX**	**M**	**W**			

3e (XX%) Using forwarding, show the 6-stage pipeline sequence (IF, IS, ID, EX, M, WB)
and draw lines showing the forwarding.

Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
sw $1, 4($2)	**IF**	**IS**	**ID**	**EX**	**M**	**W**										
sub **$4**, $1, $2		**IF**	**IS**	**ID**	**EX**	**M**	**W**									
add **$5**, **$4**, $2			**IF**	**IS**	**ID**	**EX**	**M**	**W**								
lw $3, 8(**$5**)				**IF**	**IS**	**ID**	**EX**	**M**	**W**							

3f (XX%) Increase the number of pipeline stages to **7** stages. What stage would you split and why?

I would split the MEM stage into 2 stages (Mem1=4ns and Mem2=4ns) since is the slowest resource (8ns) of all the others.
Note: any other combination would not be optimal: Mem1=3ns and Mem2=5ns

Decreasing the worst case stage delay allows for the pipeline clock to increase.

This results in a faster MIPS.

3g (XX%) What is the 7-stage pipeline clock? 4ns or 250 MHz and (assuming no hazards) MIPS= 250 MIPS

MIPS = Clock/CPI = 250 Mhz / 1CPI
Note: 250 MIPS is now faster than all cases in problem 2b!

Problem 4. Show all calculations for the following questions.
Assume an add takes 1 cycle if no dependency and if dependant then 3 clocks
Assume there is a 30% data dependency.
Assume a branch takes 2 cycle if true prediction and if false prediction then 7 clocks.
Assume that 15% of the branches are mispredicted.

4a (XX%) What is the average add instruction time in clocks? __**1.6**___

1*70% + 3*30% = 1.6

4b (XX%) What is the average branch instruction time in clocks? ___**2.75**___

2*85% + 7*15% = 2.75