

NAME: SOLUTIONS

EGRE 426

Quiz 2

Open book / Open notes

November 10, 2009

1. On the MIPS control unit description shown below; fill in the blanks to show how to add the lui instruction. You may add new microinstructions if they are necessary and reasonable. Use a dash where no new operation is necessary.

lui	rt,n	$rt \leftarrow n \parallel 0^{16}$
-----	------	------------------------------------

THE MIPS CONTROL UNIT V 5.03

Needs to be MODIFIED for SW preceded by LW.

IF:	- / $IR \leftarrow IM(PC)$, - / $PC \leftarrow PC + 4$
ID:	not J / $PCX \leftarrow PC$, - / $IRX \leftarrow IR$, - / $A \leftarrow GPR(IR.Rs)$, - / $B \leftarrow GPR(IR.Rt)$
EX:	<p>IF $IRX.Rs = IRM.Rd$ THEN $\alpha := ALUM$ ELSE IF $IRX.Rs = IRW.Rd$ THEN $\alpha := ALUW$ ELSE $\alpha := A$ IF $IRX.Rt = IRM.Rd$ THEN $\beta := ALUM$ ELSE IF $IRX.Rt = IRW.Rd$ THEN $\beta := ALUW$ ELSE $\beta := B$ $(IRX.OP = ADD) / ALUM \leftarrow \alpha + \beta$ $(IRX.OP = AND) / ALUM \leftarrow \alpha \& \beta$ $(IRX.OP = ADDI) / ALUM \leftarrow \alpha + \pm IRX.n$ $(IRX.OP = ANDI) / ALUM \leftarrow \alpha \& \pm IRX.n$ $(IRX.OP = SW) / ALUM \leftarrow \alpha + \pm IRX.n$, $SMDR \leftarrow \beta$ $(IRX.OP = LW) / ALUM \leftarrow \alpha + \pm IRX.n$ $(IRX.OP = J) / PC \leftarrow PCX(31..28) \parallel IRX.addr \parallel 00_2$ $(IRX.OP = JR) / PC \leftarrow \alpha$ $(IRX.OP = SLT) \& (\alpha < \beta) / ALUM \leftarrow 1$ $(IRX.OP = SLT) \& (\alpha \geq \beta) / ALUM \leftarrow 0$ $(IRX.OP = BEQ) \& (\alpha = \beta) / PC \leftarrow PCX + \pm IRX.n \parallel 00$, $ZERO \leftarrow 1$ $(IRX.OP = BEQ) \& (\alpha \neq \beta) / ZERO \leftarrow 0$ $(IRX.OP = LUI) / ALUM \leftarrow IRX.n \parallel 0^{16}$ ____ - / $IRM \leftarrow IRX$ Note: Loading PC in this stage disables $PC \leftarrow PC + 4$ in IF stage.</p>
DM:	<p>$(IRM.OP \neq LW)$ / $ALUW \leftarrow ALUM$, - / $IRW \leftarrow IRM$ $(IRM.OP = SW) / M(ALUM) \leftarrow SMDR$ $(IRM.OP = LW) / ALUW \leftarrow M(ALUM)$ $(IRM.OP = J) / -$ $(IRM.OP = JR) / -$ $(IRM.OP = BEQ) \& ZERO / -$ $(IRM.OP = LUI) / -$</p>
WB:	<p>$(IRW.OP = ADD) / GPR(IRW.Rd) \downarrow ALUW$ $(IRW.OP = AND) / GPR(IRW.Rd) \downarrow ALUW$ $(IRW.OP = ADDI) / GPR(IRW.Rt) \downarrow ALUW$ $(IRW.OP = ANDI) / GPR(IRW.Rt) \downarrow ALUW$ $(IRX.OP = SLT) / GPR(IRW.Rd) \downarrow ALUW$ $(IRW.OP = LW) / GPR(IRW.Rt) \downarrow ALUW$ $(IRW.OP = SW \text{ or } J \text{ or } JR \text{ or } BEQ) / -$ $(IRW.OP = LUI) / GPR(IRW.Rt) \downarrow ALUW$ NOTE: $X \downarrow Y$ means that Y is transferred into X on the clock edge that occurs before the edge that causes $A \leftarrow B$.</p>

2 On the MIPS control unit description shown below, fill in the blanks to show how to add the slti instruction. You may add new microinstructions if they are necessary and reasonable. Use a dash where no new operation is necessary.

slti	rt,rs,n	if $rs < \pm n^1$ then $rt \leftarrow 1$ else $rt \leftarrow 0$
------	---------	---

THE MIPS CONTROL UNIT V 5.03

IF:	- / $IR \leftarrow IM(PC)$, - / $PC \leftarrow PC + 4$
ID:	not J / $PCX \leftarrow PC$, - / $IRX \leftarrow IR$, - / $A \leftarrow GPR(IR.Rs)$, - / $B \leftarrow GPR(IR.Rt)$
EX:	<p>IF $IRX.Rs = IRM.Rd$ THEN $\alpha := ALUM$ ELSE IF $IRX.Rs = IRW.Rd$ THEN $\alpha := ALUW$ ELSE $\alpha := A$ IF $IRX.Rt = IRM.Rd$ THEN $\beta := ALUM$ ELSE IF $IRX.Rt = IRW.Rd$ THEN $\beta := ALUW$ ELSE $\beta := B$ $(IRX.OP = ADD) / ALUM \leftarrow \alpha + \beta$. . . $(IRX.OP = LW) / ALUM \leftarrow \alpha + \pm IRX.n$ $(IRX.OP = J) / PC \leftarrow PCX(31..28) IRX.addr 00_2$ $(IRX.OP = JR) / PC \leftarrow \alpha$ $(IRX.OP = SLT) \& (\alpha < \beta) / ALUM \leftarrow 1$ $(IRX.OP = SLT) \& (\alpha \geq \beta) / ALUM \leftarrow 0$ $(IRX.OP = BEQ) \& (\alpha = \beta) / PC \leftarrow PCX + \pm IRX.n 00$, $ZERO \leftarrow 1$ $(IRX.OP = BEQ) \& (\alpha \neq \beta) / ZERO \leftarrow 0$ $(IRX.OP = SLTI) \& (\alpha < \pm IRX.n) / ALUM \leftarrow 1$ $(IRX.OP = SLTI) \& (\alpha \geq \pm IRX.n) / ALUM \leftarrow 0$ - / $IRM \leftarrow IRX$</p> <p>Note: Loading PC in this stage disables $PC \leftarrow PC + 4$ in IF stage.</p>
DM:	<p>$(IRM.OP \neq LW) / ALUW \leftarrow ALUM$, - / $IRW \leftarrow IRM$ $(IRM.OP = SW) / M(ALUM) \leftarrow SMDR$ $(IRM.OP = LW) / ALUW \leftarrow M(ALUM)$ $(IRM.OP = J) / -$ $(IRM.OP = JR) / -$ $(IRM.OP = BEQ) \& ZERO / -$ $(IRM.OP = SLTI) / -$</p>
WB:	<p>$(IRW.OP = ADD) / GPR(IRW.Rd) \downarrow ALUW$ $(IRW.OP = AND) / GPR(IRW.Rd) \downarrow ALUW$ $(IRW.OP = ADDI) / GPR(IRW.Rt) \downarrow ALUW$ $(IRW.OP = ANDI) / GPR(IRW.Rt) \downarrow ALUW$ $(IRX.OP = SLT) / GPR(IRW.Rd) \downarrow ALUW$ $(IRW.OP = LW) / GPR(IRW.Rt) \downarrow ALUW$ $(IRW.OP = SW \text{ or } J \text{ or } JR \text{ or } BEQ) / -$ $(IRW.OP = SLTI) / GPR(IRW.Rd) \downarrow ALUW$ NOTE: $X \downarrow Y$ means that Y is transferred into X on the clock edge that occurs before the edge that causes $A \leftarrow B$.</p>

¹ rs and $\pm n$ are treated as signed integers.

3. For each instruction, enter the correct register address under the rs, rt, and rd in the table.

	31	6	5	5	5	5	6	0
	Opcode	rs	rt	rd	shamt	func		
ADD \$5,\$6,\$7	0x00	6	7	5	0x00	0x20		
	31	6	5	5	16			0
	Opcode	rs	rt	n/offset				
BEQ \$6,\$7,0x5000	0x04	6	7	0x5000				
	31	6	5	5	16			0
	Opcode	rs	rt	n/offset				
ADDI \$6,\$5,0x7fff	0x08	5	6	0x7FFF				
	31	6	5	5	5	6		0
	Opcode	rs	rt	rd	shamt	func		
ADD \$7,\$6,\$5	0x00	6	5	7	0x00	0x20		

4. The code below executes on a MIPS with correctly implemented forwarding. The initial values contained in registers \$5, \$6 and \$7 are 5, 6 and 7 respectively.

```
add  $5,$6,$7
beq  $6,$7,0x5000
addi $6,$5,0x7fff
add  $7,$6,$5
```

Using hexadecimal values, fill in the table below to show the values in registers \$5, \$6 and \$7. You do not need to show leading 0's.

Condition	\$5	\$6	\$7
Initial conditions	0x00000005	0x00000006	0x00000007
After add \$5,\$6,\$7 exits WB	0x0000000D	0x00000006	0x00000007
After addi \$6,\$5,0x7fff exits WB	0x0000000D	0x0000800C	0x00000007
After add \$7,\$6,\$5 exits WB	0x0000000D	0x0000800C	0x00008019

5. The forwarding rules as given in the EX stage of “THE MIPS CONTROL UNIT V 5.03” are shown below.

```
IF IRX.Rs = IRM.Rd THEN  $\alpha$  := ALUM
ELSE
    IF IRX.Rs = IRW.Rd THEN  $\alpha$  := ALUW ELSE  $\alpha$  := A
IF IRX.Rt = IRM.Rd THEN  $\beta$  := ALUM
ELSE
    IF IRX.Rt = IRW.Rd THEN  $\beta$  := ALUW ELSE  $\beta$  := B
```

These rules are incomplete and oversimplified. They must be expanded to work in general.

a). Explain what extensions would be necessary to insure that the forwarding always works correctly when an “add” instruction is being executed.

ANS: Do not forward from ALUM or ALUW unless instruction in DM or WB will actually load a new value into IRX.Rs or IRX.Rt²; therefore, must check that instruction in the DM or WB state is an R_type instruction. Don’t forward if \$0 is the destination register in DM or WB stage.

b). Explain what extensions would be necessary to insure that the forwarding always works correctly when an “addi” instruction is being executed.

ANS: For addi the destination register must be changed from IRX.Rs to IRX.Rt.

6. The loop shown below is executed many times. Assume the code is executed on the MIPS described by THE MIPS CONTROL UNIT V 5.03 where any problems with the α and β forwarding have been corrected and delayed branches are not used.

```
Loop: lw    $2, 0($10)
      sub   $4, $2, $3
      sw    $4, 0($10)
      addi  $10, $10, 4
      bne   $10, $30, Loop
```

Each time through the loop would require five clock cycles if it were not for stalls and aborts. How many clock cycles are actually required each time through the loop assuming the branch is taken.

ANS: lw sub one stall bne two aborts.

Therefore, total time = 5 + 1 + 2 = 8

² For example suppose the instruction in the DM stage is “beq \$6,\$7,0x5000” Then IRM.Rd = 5, but the beq does not load a new value into \$5.

7. Suppose the bne in problem 6 is changed to a delayed branch of two, and the code is rewritten as shown below. Does the rewritten code produce the same result as the original code? If not explain why not.

```

Loop: lw    $2, 0($10)
      addi   $10,$10,4
      bne    $10, $30, Loop
      sub     $4, $2, $3
      sw     $4, -4($10)

```

ANS: Yes. This produces the same result as the code in problem 6. The only difference is that it executes faster.

8. If the original loop in problem 6 is known to be a multiple of two it can be unrolled once as shown below:

	Original code	
Loop:	lw	\$2, 0(\$10)
	Sub	\$4,\$2,\$3
	Sw	\$4,0(\$10)
	Lw	\$5,4(\$10)
	Sub	\$6,\$5,\$3
	Sw	\$6,4(\$10)
	Addi	\$10,\$10,8
	Bne	\$10,\$30,loop

Assume the unrolled code is executed on the MIPS described by THE MIPS CONTROL UNIT V 5.03. Any problems with the α and β forwarding have been corrected and delayed branches are not used. Each time through the loop would require eight clock cycles if it were not for stalls and aborts. How many clock cycles are actually required each time through the loop assuming the branch is taken.

ANS: Each lw-sub causes one stall. The bne, when taken, causes two aborts. Therefore, total time = 8 + 1 + 1 + 2 = 12.

9. Fill in the table at the right to show how to reschedule (i.e. rearrange) the code in problem 8 to execute in the fastest possible way. Do not change the number of instructions!

	Original code		Scheduled code for improved performance	
Loop:	lw	\$2, 0(\$10)	lw	\$2, 0(\$10)
	Sub	\$4,\$2,\$3	Lw	\$5,4(\$10)
	Sw	\$4,0(\$10)	Sub	\$4,\$2,\$3
	Lw	\$5,4(\$10)	Sub	\$6,\$5,\$3
	Sub	\$6,\$5,\$3	Sw	\$4,0(\$10)
	Sw	\$6,4(\$10)	Sw	\$6,4(\$10)
	Addi	\$10,\$10,8	Addi	\$10,\$10,8
	Bne	\$10,\$30,loop	Bne	\$10,\$30,loop