MASSACHUSETTS INSTITUTE OF TECHNOLOGY
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

6.004 Computation Structures
Fall 2003

Quiz #4: November 14, 2003

Name DAN PORTS	*Athena Username* drkp	*Score* 5+8+6+5=24
☐ WF 10, 34-304 Morris	☐ WF 12, 34-302 Fitzpatrick	☐ WF 1, 34-304 Asanovic
☐ WF 11, 34-303 Morris	☒ WF 12, 34-303 Mazzola Paluska	☐ WF 2, 34-303 Asanovic
☐ WF 11, 36-372 Fitzpatrick	☐ WF 1, 34-303 Mazzola Paluska	

NOTE: You'll find reference sheets for the Beta (instruction set summary, unpipelined Beta, 5-stage Beta) and some scratch pipeline diagrams at the end of the Quiz.

5 **Problem 1.** (6 Points)

(A) The instruction **`BR(tag)`** is stored at location 0x1000. You examine the coded instruction word in memory and find that its literal portion is zero. What hex location will the instruction branch to?

Target hex address for BR: 0x 1004

(B) If an application uses the XP register, this may prevent interrupts from returning to the proper location. If the XP reg is used in the interrupt handler, then it may return to the wrong place. **Circle one: TRUE .. FALSE** -1

(C) The pipelined Beta's occasional need to stall following a LD instruction could be eliminated by additional bypass paths in the processor's datapath.

Circle one: TRUE .. FALSE

(D) An exception (e.g., an invalid instruction address) detected in the instruction fetch stage of a pipelined Beta will cause instructions in each of the other pipeline stages to be annulled.

Circle one: TRUE .. FALSE

(E) Both the pipelined and unpipelined Betas described in lecture execute instructions at about one CPI (clocks per instruction).

Circle one: TRUE ... FALSE

(F) Ben Bitdiddle is thinking about modifying a 5-stage pipelined Beta to add a "Jump if Memory Zero" instruction (JMZ) that fetches the contents of a memory location and jumps if the fetched value is zero. How many branch delay slots would follow a JMZ instruction in the modified 5-stage pipelined Beta?

Number of branch delay slots following JMZ: 4

IF ICF ALU MEM WB

Problem 2. (8 Points)

Marketing has asked for the following instructions to be added to an Extended Beta instruction set, for implementation on an *unpipelined* Beta:

```
LDX( Ra, Rb, Rc )          // Load indexed
  EA ← Reg[Ra] + Reg[Rb]
  Reg[Rc] ← Mem[EA]
  PC ← PC + 4

SNZ(Rc, C, Ra)             // Store if non-zero
  EA ← Reg[Ra]+SEXT(C)
  if Reg[Rc] != 0 then Mem[EA]← Reg[Rc]
  PC ← PC + 4
```

Can't implement comparison of Reg[Rc] ≠ 0.

// The following instruction assumes (as you may) that the Beta's memory can be read and
// written during a single clock cycle. The previous contents of the addressed location is
// output from the RD (read data) port, while the new data (applied to the WD inputs)
// becomes the new contents of the addressed location.

```
MSWP(Ra, C, Rc)            // Swap register with memory
  EA ← Reg[Ra] + SEXT(C)
  tmp ← Mem[EA]
  Mem[EA] ← Reg[Rc]
  Reg[Rc] ← tmp
  PC ← PC + 4
```

For each instruction either fill in the appropriate values for the control signals in the table below or put a line through the whole row if the instruction cannot be implemented using the existing unpipelined Beta datapath. Use "—" to indicate a "don't care" value for a control signal.

Instr	ALUFN	WERF	BSEL	WDSEL	WR	RA2SEL	PCSEL	ASEL	WASEL
LDX	+	1	0	2	0	0	0	0	0
~~SNZ~~									
MSWP	+	1	1	2	1	1	0	0	0

6

Problem 3. (6 Points)

Consider the execution of each of the following code sequences by a Beta implementation with a 5-stage pipeline, full bypassing and one branch delay slot with annulment (i.e., the hardware annuls the already-fetched instruction immediately following a branch when the branch is taken). When the last instruction in the sequence is in the RF stage, please indicate which bypass paths will be activated for the "Ra" and "Rb/Rc" operands by specifying one of

RF	operand fetched from register file
ALU	operand bypassed from ALU stage
MEM	operand bypassed from memory stage
WB	operand bypassed from writeback stage
NONE	operand is not a register

If the instruction is stalled in the RF stage for several cycles, please indicate the bypass paths activated on the instruction's final cycle in the RF stage.

```
ADDC(R31,17,R5)
ADD(R0,R5,R5)
SUBC(R0,4,R4)
ADDC(R0,1,R0)
MUL(R5,R4,R5)
```

(A) When MUL is in the RF stage: Ra bypass= WB **Rb/Rc bypass=** MEM

```
LD(R31,data1,R0)
LD(R31,data2,R1)
CMPEQ(R0,R1,R2)
```

(B) When CMPEQ is in the RF stage: Ra bypass= RF **Rb/Rc bypass=** WB

```
      BR(foo,LP)
      SUBC(SP,4,SP)
      ...

foo:  ADDC(SP,4,SP)      | PUSH(LP) expanded...
      ST(LP,-4,SP)       | remember how ST is encoded!
```

(C) When ST is in the RF stage: Ra bypass= ALU **Rb/Rc bypass=** WB

Problem 4. (5 Points)

45

Using a technique called "memory-mapped I/O," the registers controlling an I/O device appear as locations in the memory space of the processor where they are easy to access using regular LD and ST instructions. In the following example, the processor "busy waits" in a tight loop testing the status register of an I/O device, which will read as non-zero if there's something for the processor to do.

```
loop:  LD(R31, status, R0)    | poll the status register
       BEQ(R0, loop, R31)     | keep polling if it's zero
       ADD(R0, R1, R2)        | ahh, something useful to do
       ...

status = 0xFFFFFF00   | address of memory-mapped status registers
```

Please complete the pipeline diagram below, showing the execution of this instruction sequence on a standard 5-stage pipelined Beta with full bypassing and one branch delay slot with annulment. Assume that the status location reads as zero, i.e., the BEQ is taken.

IF	LD	BEQ	ADD	ADD ✓	ADD ✓	LD ✓
RF		LD	BEQ	BEQ ✓	BEQ ✓	NOP ✓
ALU			LD	NOP ✓	NOP ✓	BEQ ✓
MEM				LD	NOP ✓	NOP ✓
WB					LD	NOP ✓

END OF QUIZ 4.