50.002 COMPUTATIONAL STRUCTURES

INFORMATION SYSTEMS TECHNOLOGY AND DESIGN

Problem Set 10

1 Problem 1

Notta Kalew, a somewhat fumble-fingered lab assistant, has deleted the opcode field from the following table describing the control logic of an unpipelined Beta processor.

PCSEL	RA2SEL	ASEL	BSEL	WDSEL	ALUFN	WR	WERF	WASEL
0	-	0	1	1	A-B	0	1	0
Z?1:0	-	-	-	0	-	0	1	0
0	-	1	-	2	A	0	1	0
0	0	0	0	1	CMPEQ	0	1	0
0	1	0	1	-	ADD	1	0	0

Figure 1

1. Help Notta out by identifying which Beta instruction is implemented by each row of the table.

   Solution:
   From first row to the last: SUBC, BEQ, LDR, CMPEQ, ST

2. Notta notices that WASEL is always zero in this table. Explain briefly under what circumstances WASEL would be non-zero.
   Solution:
   WASEL is 1 if an interrupt, an illegal opcode is trapped, or a fault occurs. When WASEL is 1, it selects XP as the write address for the register file; Reg[XP] is where we store the current PC+4 whenever there is an interrupt, a fault, or an illegal opcode.

3. Notta has noticed the following C code fragment appears frequently in the benchmarks:

   ```
   int *p; /* Pointer to integer array */
   ```

```
int i,j; /* integer variables */
...
j = p[i]; /* access ith element of array */
```

The pointer variable p contains the address of a dynamically allocated array of integers. The value of p[i] is stored at the address Mem[p] +4*Mem[i] where p and i are locations containing the values of the corresponding C variables. On a conventional Beta this code fragment is translated to the following instruction sequence:

```
LD(...,R1)     /* R1 contains p, the array base address */
LD(...,R2)     /* R2 contains I, the array index */    ...
SHLC(R2,2,R0)  /* compute byte-addressed offset = 4*i */
ADD(R1,R0,R0)  /* address of indexed element */
LD(R0,0,R3)    /* fetch p[i] into R3 */
```

Notta proposes the addition of an LDX instruction that shortens the last three instructions to:

```
SHLC(R2,2,R0)  /* compute byte-addressed offset = 4*i */
LDX(R0,R1,R3)  /* fetch p[i] into R3 */
```

Give a register-transfer language description for the LDX instruction. Examples of register-transfer language descriptions can be for other Beta instructions in the Beta Documentation handed out in lecture.

Solution:

```
LDX( Ra, Rb, Rc )
    EA <- Reg[Ra] + Reg[Rb]
    Reg[Rc] <- Mem[EA]
    PC <- PC + 4
```

4. Using a table like the one above specify the control signals for the LDX opcode.

Solution:

PSSEL	RA2SEL	ASEL	BSEL	WDSEL	ALUFN	WR	WERF	WASEL
0	0	0	0	2	ADD	0	1	0

Table 1

5. It occurs to Notta that adding an STX instruction would probably be useful too. Using this new instruction, p[i] = j might compile into the following instruction sequence:

```
SHLC(R2,2,R0)  /* compute byte-addressed offset = 4*i */
STX(R3,R0,R1)  /* R3 contains j, R1 contains p */
```

Briefly describe what modifications to the Beta datapath would be necessary to be able to execute STX in a single cycle.

Solution:
The register transfer language description of STX would be:

```
STX(Rc, Rb, Ra)
EA <- Reg[Ra] + Reg[Rb]
Mem[EA] <- Reg[Rc]
PC <- PC + 4
```

It's evident that we need to perform 3 register reads, but the Beta's register file has only 2 read ports. Thus we need to add a third read port to the register file. Incidentally, adding a third read port would eliminate the need for the RA2SEL mux because we no longer need to choose between Rb and Rc, since each register field has its own read port.

2 Beta Short Questions

1. In an unpipelined Beta implementation, when is the signal RA2SEL set to "1"?

   Solution:
   The RA2SEL signal is set to 1 when executing a ST instruction. When RA2SEL is 1 the 5-bit Rc field of the instruction is sent to the RA2 port of the register file, causing Reg[Rc] to be sent to the write data port of main memory.

2. In an unpipelined Beta implementation, when executing a BR(foo,LP) instruction to call procedure foo, what should WDSEL should be set to?
   Solution:
   BR(foo,LP) == BEQ(R31,foo,LP). All BNE/BEQ instructions save the address of the following instruction in the specified destination register (LP in the example instruction). So WDSEL should be set 0, selecting the output of the PC+4 logic as the data to be written into the register file.

3. The minimum clock period of the unpipelined Beta implementation is determined by the propagation delays of the data path elements and the amount of

time it takes for the control signals to become valid. Which of the following select signals should become valid first in order to ensure the smallest possible clock period: PCSEL, RA2SEL, ASEL, BSEL, WDSEL, WASEL?

Solution:
To ensure the smallest possible clock period RA2SEL should become valid first. The RA2SEL mux must produce a stable register address before the register file can do its thing. All other control signals affect logic that operates after the required register values have been accessed, so they don't have to be valid until later in the cycle.