Computation Structures

Beta Implementation Worksheet

Unpipelined Beta


```
Reset: 0x80000000
Illop: 0x80000004
XAdr:  0x80000008
```
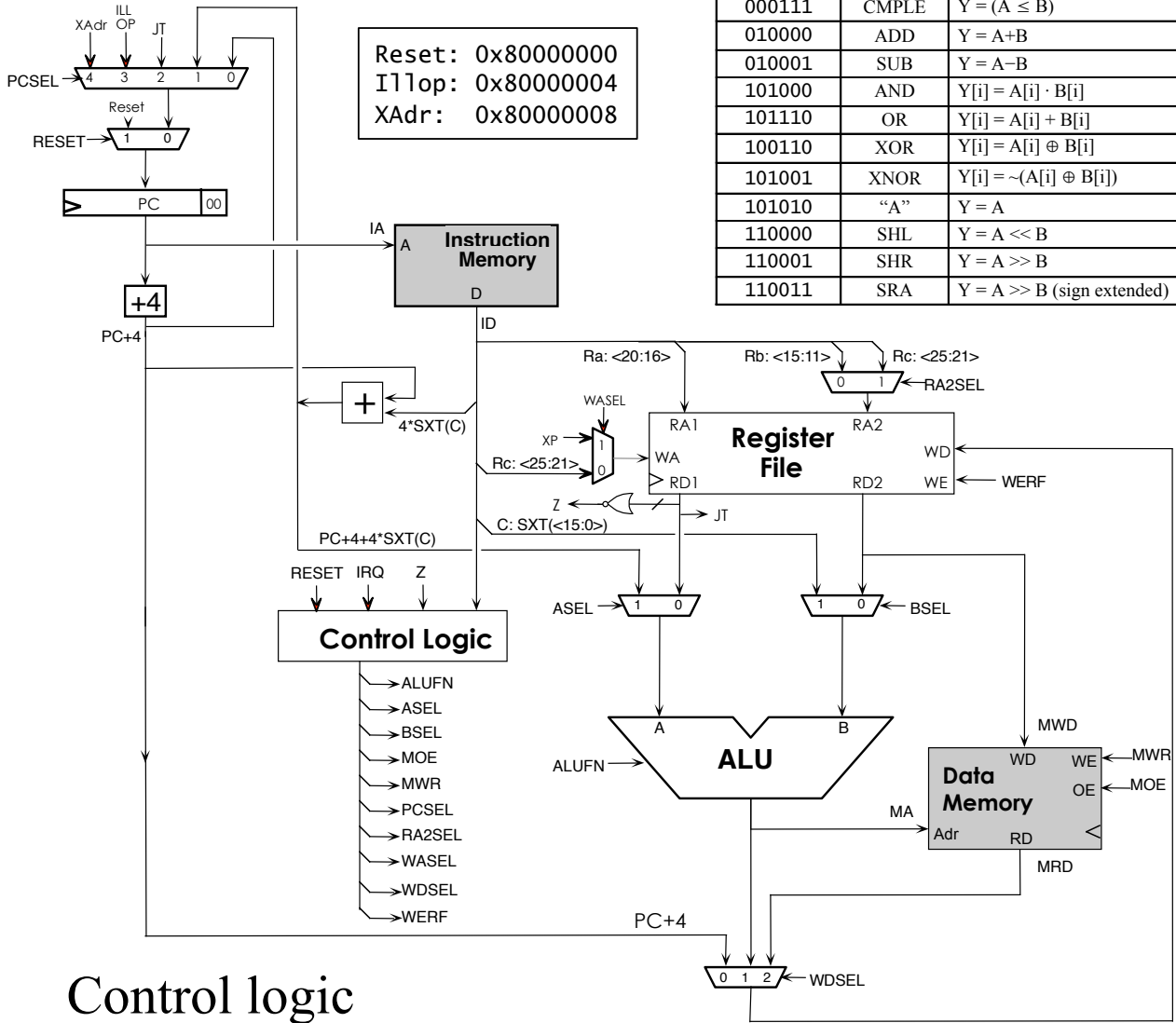
ALUFN[5:0]	Operation	Output value Y[31:0]
000011	CMPEQ	$Y = (A == B)$
000101	CMPLT	$Y = (A < B)$
000111	CMPLE	$Y = (A \leq B)$
010000	ADD	$Y = A+B$
010001	SUB	$Y = A-B$
101000	AND	$Y[i] = A[i] \cdot B[i]$
101110	OR	$Y[i] = A[i] + B[i]$
100110	XOR	$Y[i] = A[i] \oplus B[i]$
101001	XNOR	$Y[i] = \sim(A[i] \oplus B[i])$
101010	"A"	$Y = A$
110000	SHL	$Y = A << B$
110001	SHR	$Y = A >> B$
110011	SRA	$Y = A >> B$ (sign extended)

Control logic

	RESET	IRQ	OP	OPC	LD	LDR	ST	JMP	BEQ	BNE	ILLOP
ALUFN[5:0]	--	--	F(op)	F(op)	"+"	"A"	"+"	--	--	--	--
ASEL	--	--	0	0	0	1	0	--	--	--	--
BSEL	--	--	0	1	1	--	1	--	--	--	--
MOE	--	--	--	--	1	1	0	--	--	--	--
MWR	0	0	0	0	0	0	1	0	0	0	0
PCSEL[2:0]	--	4	0	0	0	0	0	2	Z ? 1 : 0	Z ? 0 : 1	3
RA2SEL	--	--	0	--	--	--	1	--	--	--	--
WASEL	--	1	0	0	0	0	--	0	0	0	1
WDSEL[1:0]	--	0	1	1	2	2	--	0	0	0	0
WERF	--	1	1	1	1	1	0	1	1	1	1

Problem 1.

For this problem assume that each register has been initialized to the value 0x0000??00 where "??" is the register number as a two-digit hex number. So R0 is initialized to 0x00000000, R1 to 0x00000100, …, and R30 to 0x00001E00. R31 of course always reads as 0.

For each instruction below, please indicate the values that will be found in the unpipelined Beta datapath just before the end of the clock cycle in which the instruction is executed. If the value doesn't matter since it's not used during the execution of the instruction or can't be determined, write "−".
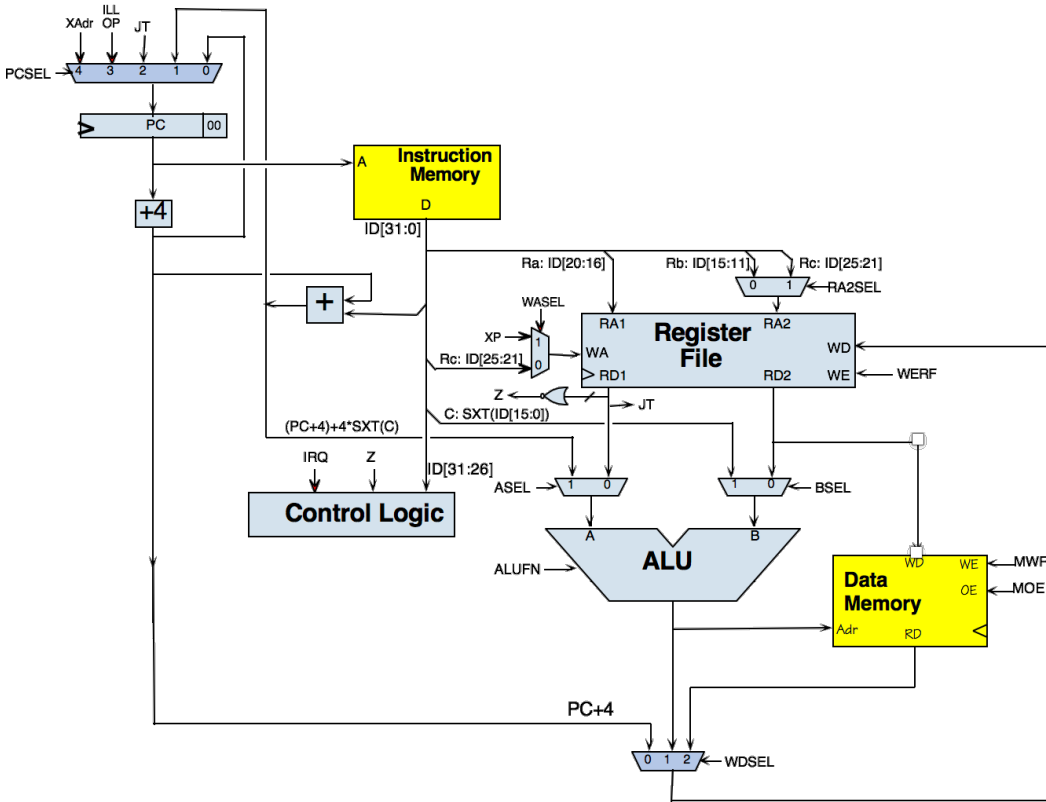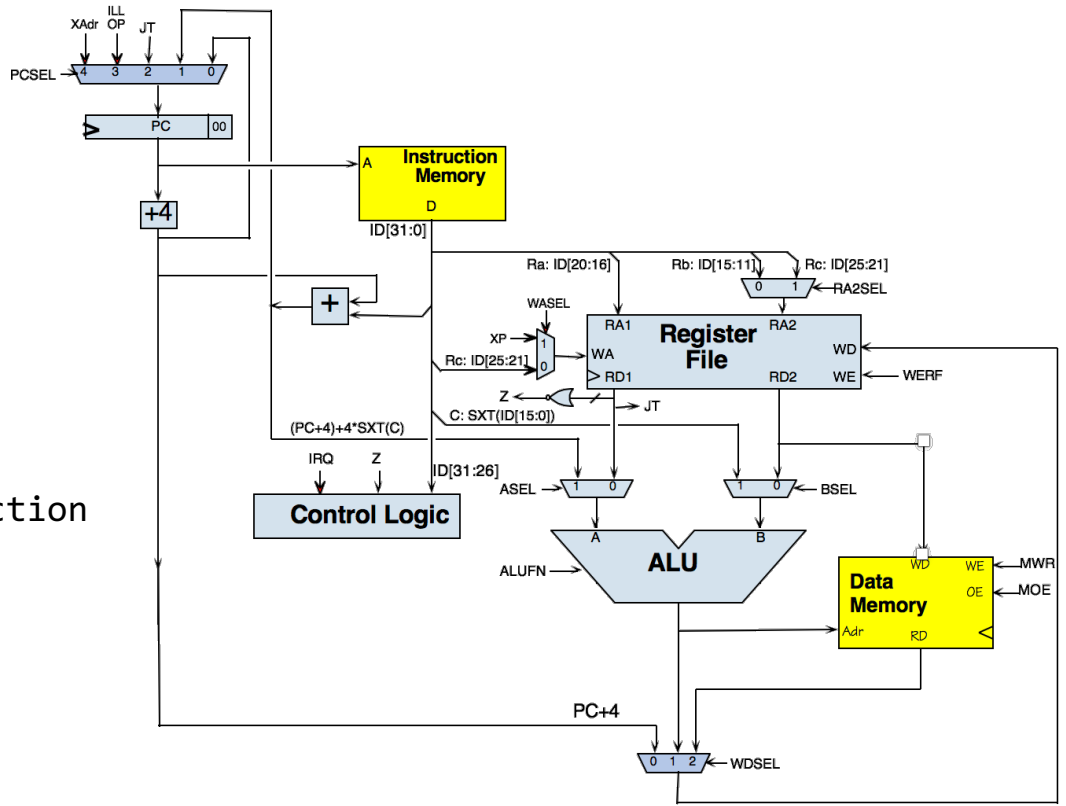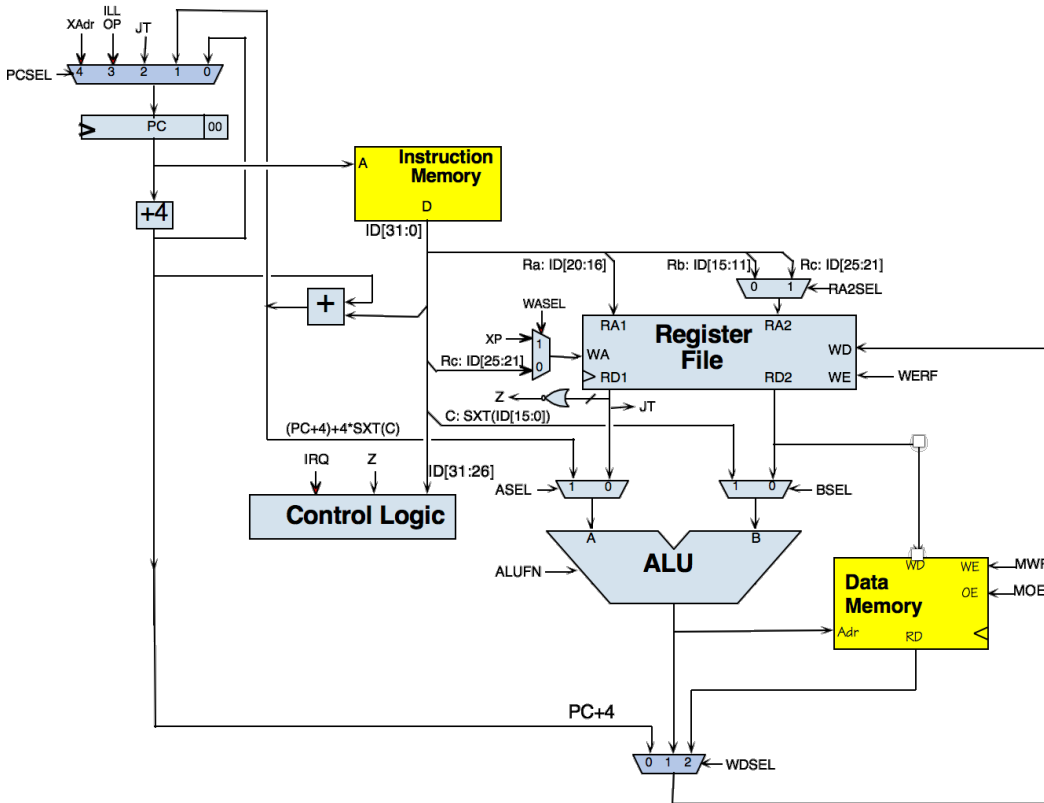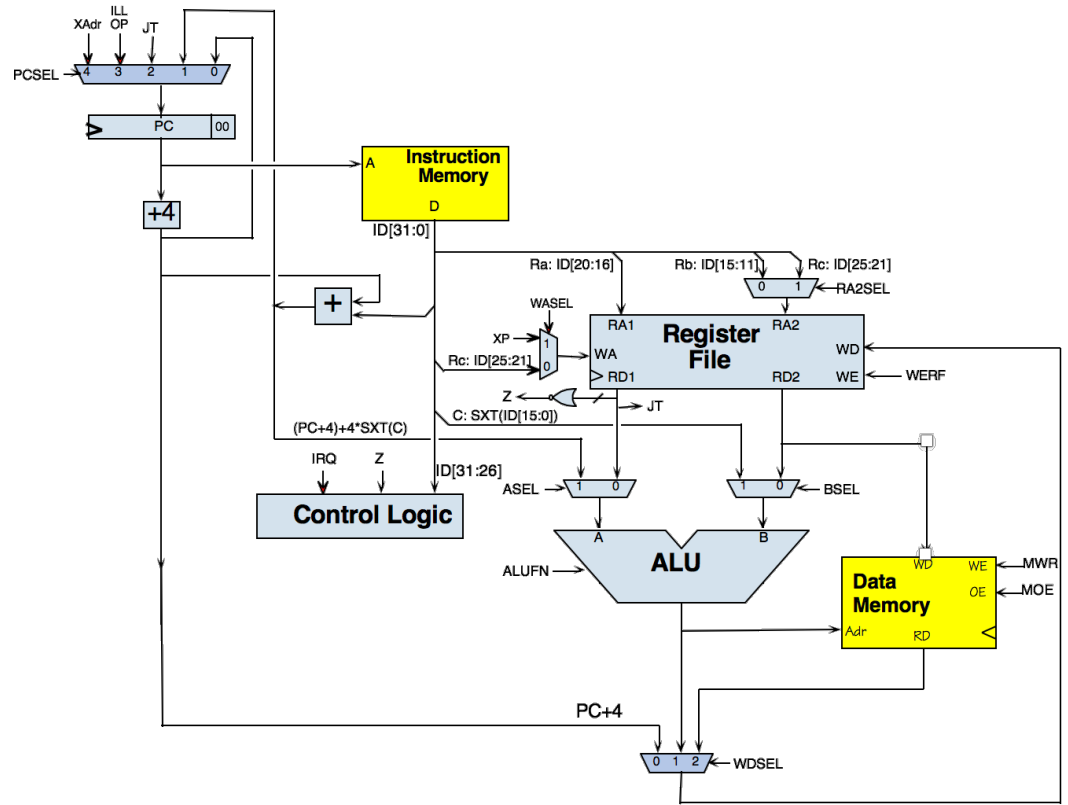
. = 0x100
SUB(R5,R3,R7)
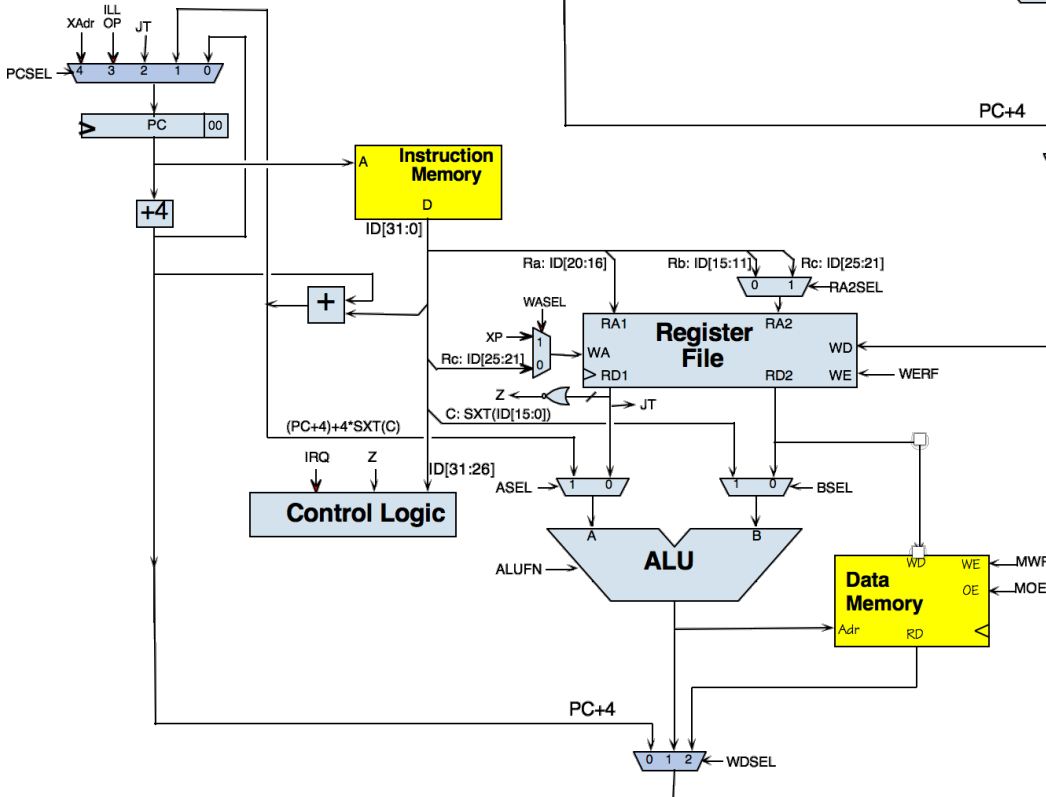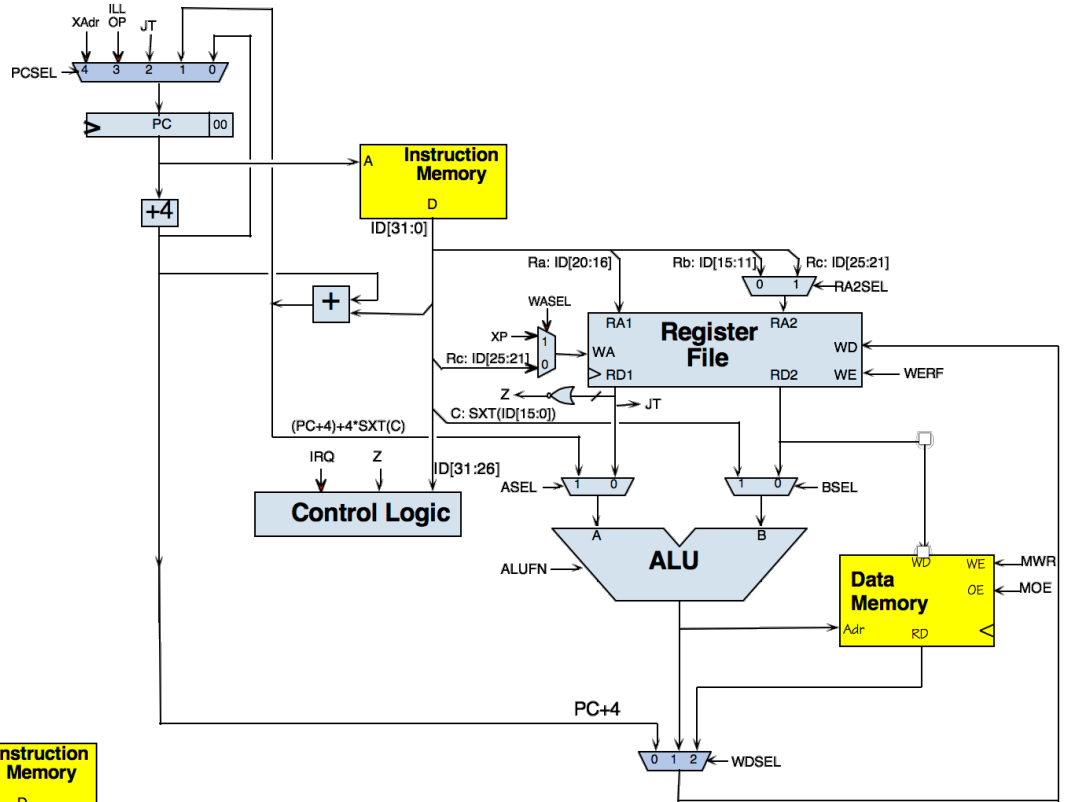
. = 0x100
SHLC(R30,8,R16)

. = 0x100
LD(R3,-0x200,R7)

// hex for instruction
0x60E3FE00

. = 0x100
ST(R3,-0x200,R7)

. = 0x100
JMP(LP)

. = 0x100
BEQ(R31,.+0x80,LP)

Problem 2.

Consider adding the following instructions to the Beta instruction set, for implementation on the Beta hardware shown in lecture (see diagram included in the reference material at the end of this quiz). You're allowed to change how the control signals are generated but no modifications to the datapath are permitted.

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 Beta datapath. Use "—" to indicate a "don't care" value for a control signal. The values can be a function of Z (which is 1 when Reg[Ra] is zero).

> **LDX(Ra, Rb, Rc)**   // Load indexed
>   EA ← Reg[Ra] + Reg[Rb]
>   Reg[Rc] ← Mem[EA]
>   PC ← PC + 4
>
> **STX(Ra, Rb, Rc)**   // Store indexed
>   EA ← Reg[Ra] + Reg[Rb]
>   Mem[EA] ← Reg[Rc]
>   PC ← PC + 4
>
> **MVZC(Ra, literal, Rc)**   // Move constant if zero
>   If Reg[Ra] == 0 then Reg[Rc] ← SXT(literal)
>   PC ← PC + 4
>
> **SOB(Ra, literal, Rc)**   // Subtract one and branch
>   PC ← PC + 4
>   EA ← PC + 4*SEXT(literal)
>   tmp ← Reg[Ra]
>   Reg[Rc] ← Reg[Ra] – 1
>   if tmp != 0 then PC ← EA
>
> **ARA(Ra, literal, Rc)**   // Add Relative Address
>   Reg[Rc] ← Reg[Rc] + PC + 4 + 4*SEXT(literal)
>   PC ← PC + 4

(FILL IN TABLE BELOW)

Instr	ALUFN	WERF	BSEL	WDSEL	MOE	MWR	RA2SEL	PCSEL	ASEL	WASEL
LDX										
STX										
MVZC										
SOB										
ARA										

Problem 3.

Ben Bitdiddle is proposing the short assembly language program shown to the right as a manufacturing test to ensure the correct operation of the Control ROM. He is assuming – and you may too – that the Beta datapath components (e.g., Memories, ALU, muxes, register file, adders) are working correctly and that any errors in execution are due to faulty signals from the Control ROM. Ben's plan is to run the program then look at the value in the memory location labeled ANS. If the value is 0x6004, the test passes, otherwise the Beta being tested is declared faulty and discarded.

```
 .=0
Test: LD(R31,X,R0)
      ADDC(R0,1,R1)
      BNE(R1,L1,R31)
      ADDC(R1,1,R1)
L1:   ST(R1,ANS,R31)
      HALT()
X:    .LONG(0x6003)
ANS:  .LONG(0)
```

For each of the following faults, indicate the value that the faulty Beta will store into ANS.

(A) RA2SEL is stuck at the value 0.

Value stored in ANS by faulty Beta: _____

(B) WDSEL[1:0] is stuck at the binary value 00.

Value stored in ANS by faulty Beta: _____

(C) PCSEL[2:0] is stuck at the binary value 000.

Value stored in ANS by faulty Beta: _____

Problem 4. Beta Implementation

Consider the assembly language program shown to the right. Assume that all register values are initialized to 0, execution starts at PC=0 and halts when HALT() is executed.

This program is run on 4 different broken Betas, where each Beta has a specified control signal stuck at the specified value, i.e., the control signal value is fixed and is not affected by the value produced by the Beta's CTL module.   For each broken Beta, please give the value in registers R1, R2, R3, and the location X: after the programs halts. **Assume that any don't care control signal values are 0.**

```
 . = 0
 LD(R31,X,R1)
 CMPLTC(R1,0,R2)
 BF(R2,end,R3)
 SUB(R31,R1,R1)
 ST(R1,X,R31)
END: HALT()
X:   LONG(-42)
```

Broken control signal	Final value in			
	R1	*R2*	*R3*	*Location X:*
RA2SEL stuck at 0				
WDSEL stuck at 0b00				
WASEL stuck at 1				
WERF stuck at 1				

Problem 5.

In this problem, you will consider a number of plausible hardware faults in an otherwise working Beta processor; you may want to consult the diagram and documentation on the backs of pages of this quiz. Each of the faults involves changing a particular output of the control logic to some new (incorrect) constant value. In each case, you are to evaluate the impact of the fault on each of the following Beta instructions:

```
I1:     ST(R0, 0x100, R1)
I2:     JMP(LP, R31)
I3:     BEQ(R31, .+4, R0)
I4:     SUB(R1, R0, R0)
```

For each of the following faults, identify which (if any) of the above instructions will fail to work properly – that is, if the fault might effect the processor state (register and PC values) after the execution of the instruction. Be careful: some of these are tricky!

(A) ALUFN stuck at code for "-" (32-bit SUBTRACT)

      Which instruction(s) fail? Circle all applicable, or NONE: I1 I2 I3 I4 NONE

(B) RA2SEL stuck at 1

      Which instruction(s) fail? Circle all applicable, or NONE: I1 I2 I3 I4 NONE

(C) WERF stuck at 0

      Which instruction(s) fail? Circle all applicable, or NONE: I1 I2 I3 I4 NONE

(D) BSEL stuck at 0

      Which instruction(s) fail? Circle all applicable, or NONE: I1 I2 I3 I4 NONE

Problem 6.

(A) The Beta executes the assembly program below starting at location 0 and stopping when it reaches the HALT() instruction. Please give the values in the indicated registers after the Beta stops. Write the values in hex or write "CAN'T TELL" if the values cannot be determined.

```
    . = 0
    LD(r31, X, r0)
    CMPLE(r0, r31, r1)
    BNE(r1, L1, r2)
    ADDC(r31, 1, r0)
L1: HALT()

X:  LONG(0x87654321)
```

Value left in R0 or "CAN'T TELL": 0x_____

Value left in R1 or "CAN'T TELL": 0x_____

Value left in R2 or "CAN'T TELL": 0x_____

(B) Redo part (A) but this time assume that all the control signals going to the datapath from the control logic are stuck at logic 0, *except for WERF* which operates as expected. Note that when ALUFN[4:0] = 0b00000, the ALU computes A+B.

```
        . = 0
        LD(r31, X, r0)
        CMPLE(r0, r31, r1)
        BNE(r1, L1, r2)
        ADDC(r31, 1, r0)
   L1:  HALT()

   X:   LONG(0x87654321)
```

Value left in R0 or "CAN'T TELL": 0x_____

Value left in R1 or "CAN'T TELL": 0x_____

Value left in R2 or "CAN'T TELL": 0x_____

(C) Bettah Beta Inc. (you can tell they're based in Boston!) is proposing a new Beta instruction TCLR that sets Rc to the current value of a memory location whose address is in Ra and writes a zero to that location, all in a single cycle. They are assuming that main memory works as it does in JSim: its read ports are combinational and the write port takes a CLK signal and performs the write at the end of the current cycle – *so the same memory location can be read and written in the same clock cycle*.

Here's their draft entry for the Beta reference manual:

Usage: TCLR(Ra,Rc)

Opcode:

011010	Rc	Ra	11111	*unused*

Operation: PC ← PC + 4
             EA ← Reg[Ra]
             Reg[Rc] ← Mem[EA]
             Mem[EA] ← 0

The contents of register Rc are set to the contents of the memory location whose address is in Ra. Then, at the end of the cycle, that memory location is set to 0.

Please fill in the appropriate values for the control signals that will cause the datapath to implement the correct operations OR briefly explain why TCLR cannot be implemented with the existing Beta datapath in a single cycle.

Fill in table:

Instr	ALUFN	WERF	BSEL	WDSEL	MOE	MWR	RA2SEL	PCSEL	ASEL	WASEL
TCLR										