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
SUBC ←	0	-	0	1	1	A-B	0	1	0
BNE ←	Z?1:0	-	-	-	0	-	0	1	0
LDR ←	0	-	1	-	2	A	0	1	0
CMPEQ ←	0	0	0	0	1	CMPEQ	0	1	0
ST ←	0	1	0	1	-	ADD	1	0	0

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

B. Notta notices that WASEL is always zero in this table. Explain briefly under what circumstances WASEL would be non-zero.

When IRQ = 1

C. 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 */
```

Reg[R3] ← Mem[Reg[R0] + Reg[R1]]

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.

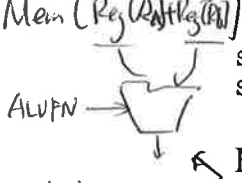
D. Using a table like the one above specify the control signals for the LDX opcode.

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

Change constant to RB

E. 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:

Load Reg[Rc] into Mem[Reg[Ra]+Reg[Rb]]




```
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.

Rc must be read in the same cycle as Ra/Rb. Therefore Regfile has three RA/RD

Problem 2. Beta quickies:

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

B. In an unpipelined Beta implementation, when executing a BR(foo,LP) instruction to call procedure foo, what should WDSEL should be set to? → BEQ ∴ WDSEL 0

C. 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?

~~WASEL~~
RA2SEL → Regfile → BSEL →
(read)
ASEL → WDSEL → Regfile (write)
WASEL

PCSEL → PC (write)

(RA2SEL) must first.