Project Description

In this project, you will design and implement a pipelined processor. The processor uses RISC-like instruction set. The processor has four internal registers: R_0, R_1, R_2, and R_3. Each register is 1-byte. The address space of instruction memory and data memory is 256, and the processor uses little-endian byte ordering. The length of all instructions is the same and is 2-byte. The instructions set of the processor is as follows:

Instructions Set
We use a RISC-like instruction set in this project. There are four 1-byte general purpose registers; R_0, R_2, R_2, and R_3. A special register which is called Link Register (LR) is used in BR.SUB and RETURN instructions. Instruction memory and data memory are separate, and the address space of each of the two memories is 256. The memories are byte addressable.
For extra credit, there are two optional instructions: BR.SUB and RETURN. BR.SUB instruction is used for subroutine call. RETURN instruction is used at the end of subroutines and changes the flow of program to the main program.

There are 3 different instruction formats:

1) A-Format
Figure 1 depicts an a-format instruction. These instructions are two bytes. The second byte is unused. In the first byte, the Op-code is the high order nibble, and the low order nibble specifies two registers.

7 … 4	3 … 2	1 … 0
op-code	ra	rb
unused		

Figure 1. A-format instructions.

Table I shows op-code values for a-format instructions and explains their functionality. R[ra] and R[rb] indicate values for registers ra and rb. There are two flags: zero flag(Z) and negative flag (N). Arithmetic instructions affect the z-flag and the n-flag. If the result of an arithmetic operation is zero/negative, the Z/N flag is set to one; otherwise, the flags are set to zero. The processor has 1-byte input port and 1-byte output port. The ports are connected to the external pins. IN and OUT instructions transfer values between the processor ports and the internal registers.

Example
For *ADD r2, r1* instruction, the op-code, ra, and rb are as follows:

op-code = 1
ra = 2
rb = 1

The bit stream for the instruction is: 00011001. Hence, the hexadecimal format of the instruction in is: 0x19

Table I. A-format instructions.

Mnemonic	Op-code	Function
NOP	0	Nothing

ADD	1	R[ra] $\leftarrow$ R[ra] + R[rb];
		((R[ra] + R[rb]) = 0) $\Rightarrow$ Z $\leftarrow$ 1; else $\Rightarrow$ Z $\leftarrow$ 0;
		((R[ra] + R[rb]) < 0) $\Rightarrow$ N $\leftarrow$ 1; else $\Rightarrow$ N $\leftarrow$ 0;

SUB	2	R[ra] $\leftarrow$ R[ra] – R[rb];
		((R[ra] – R[rb]) = 0) $\Rightarrow$ Z $\leftarrow$ 1; else $\Rightarrow$ Z $\leftarrow$ 0;
		((R[ra] – R[rb]) < 0) $\Rightarrow$ N $\leftarrow$ 1; else $\Rightarrow$N $\leftarrow$ 0;

NAND	3	R[ra] $\leftarrow$ R[ra] NAND R[rb];
		((R[ra] NAND R[rb]) = 0) $\Rightarrow$Z $\leftarrow$ 1; else $\Rightarrow$Z $\leftarrow$ 0;
		((R[ra] NAND R[rb]) < 0) $\Rightarrow$N $\leftarrow$ 1; else $\Rightarrow$N $\leftarrow$ 0;

SHL	4	Z $\leftarrow$ R[ra]<7>; R[ra] $\leftarrow$ (R[ra]<6:0>&0);
SHR	5	Z $\leftarrow$ R[ra]<0>; R[ra] $\leftarrow$ (0&R[ra]<7:1>);
OUT	6	OUT.PORT $\leftarrow$ R[ra];
IN	7	R[ra] $\leftarrow$ IN.PORT;
MOV	8	R[ra] $\leftarrow$ R[rb];

2) B-Format

B-format instructions are two bytes and are used for branch instructions. As figure 2 shows, a b-format instruction has an op-code, brx, and effective address (ea) fields. The op-code of BR.Z and BR.N is the same, and brx field is used to distinguish the two instructions. The ea holds destination address in B-format instructions. The three low order bits of the first byte is unused

7 4	3	2 0
op-code	brx	-
ea		

Figure 2. B-format instructions.

Table II shows more details for a b-format instruction. BR instruction jumps into the destination address (ea). BR.Z is a conditional branch. If the Z flag is one, it jumps into the destination. Similarly, BR.N jumps into destination address if the N flag is one.

Optional Instructions

BR.SUB is used for the subroutine call. It saves the address of the next instruction in Link Register (LR) and jumps to the subroutine address. The second byte of instruction holds subroutine address. At the end of the subroutine, the RETURN instruction writes LR into PC. Note that LR is a special register and is separate from general purpose registers (R_0 to R_3).

Table II. B-format instructions.

Mnemonic	Op-code	Function
BR	9	PC ← ea;
BR.Z	10	(brx=0 ∩ Z=1) ⇒ PC ← ea ; (brx=0 ∩ Z=0) ⇒ PC ← PC + 2;

BR.N	10	(brx=1 ∩ N=1) ⇒ PC ← ea;
		(brx=1 ∩ N=0) ⇒ PC ← PC + 2;

BR.SUB	11	LR ← PC + 2; PC ← ea
RETURN	12	PC ← LR;

3) L-Format

L-format instructions are two bytes and are used for load/store instructions. The first byte holds the op-code and ra, and the second byte holds the address of the memory location or an immediate value. Figure 3 shows an L-format instruction. Note that in an L-format instruction, the first two low order bits of the first byte are unused.

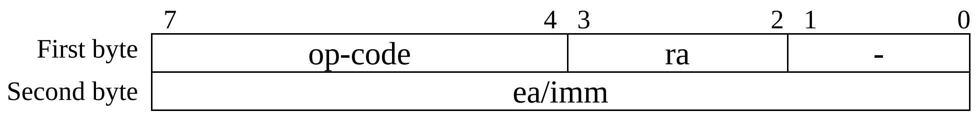

Figure 3. L-format instructions.

Table III shows more details for L-format instructions. LOAD and STORE instructions write/read register ra into/from address ea. LOADIMM writes a constant value (imm) into register ra.

Table III. L-format instructions.

Mnemonic	Op-code	Function
LOAD	13	R[ra] ← M[ea];
STORE	14	M[ea] ← R[ra];
LOADIMM	15	R[ra] ← imm;

Table IV shows summary of all instructions.

Table IV. Summary of all instructions.

Mnemonic	Op-code	Function
NOP	0	Nothing

ADD	1	R[ra] ← R[ra] + R[rb];
		((R[ra] + R[rb]) = 0) ⇒ Z ← 1; else ⇒ Z ← 0;
		((R[ra] + R[rb]) < 0) ⇒ N ← 1; else ⇒ N ← 0;

SUB	2	R[ra] ← R[ra] – R[rb];
		((R[ra] – R[rb]) = 0) ⇒ Z ← 1; else ⇒ Z ← 0;
		((R[ra] – R[rb]) < 0) ⇒ N ← 1; else ⇒N ← 0;

NAND	3	R[ra] ← R[ra] NAND R[rb];
		((R[ra] NAND R[rb]) = 0) ⇒Z ← 1; else ⇒Z ← 0;
		((R[ra] NAND R[rb]) < 0) ⇒N ← 1; else ⇒N ← 0;

SHL	4	Z ← R[ra]<7>; R[ra] ← (R[ra]<6:0>&0);
SHR	5	Z ← R[ra]<0>; R[ra] ← (0&R[ra]<7:1>);
OUT	6	OUT.PORT ← R[ra];
IN	7	R[ra] ← IN.PORT;
MOV	8	R[ra] ← R[rb];
BR	9	PC ← ea;
BR.Z	10	(brx=0 ∩ Z=1) ⇒ PC ← ea ; (brx=0 ∩ Z=0) ⇒ PC ← PC + 2;

BR.N	10	(brx=1 ∩ N=1) ⇒ PC ← ea;
		(brx=1 ∩ N=0) ⇒ PC ← PC + 2;

BR.SUB	11	LR ← PC + 2; PC ← ea
RETURN	12	PC ← LR;
LOAD	13	R[ra] ← M[ea];
STORE	14	M[ea] ← R[ra];
LOADIMM	15	R[ra] ← imm;