Project 3

Due February 25, 2018 at 11:59 PM

For this project you will be working with a partner. Note: all members of the group are not guaranteed equal credit. You will be using Logisim for this project. Submit a README.txt, and interactive grading signup with your circuit file compressed together in a tgz (tar gzip) file. Only one member of the group needs to submit the tgz file. Your README file must have your name, SID number, partner's name, partner's SID number, a brief description of what works/doesn't, and a list of sources you used for designing of your circuit (you do not need to list the book or lecture notes it is assumed these have been used). You may use any of the built in components of Logisim, except for those in the Arithmetic group. All class projects will be run through MOSS like software to determine if students have excessively collaborated. Excessive collaboration, or failure to list external sources will result in the matter being referred to Student Judicial Affairs.

You will be developing a two cycle (Fetch-Decode, Execute-Writeback) 15-bit CPU for this project. The CPU will contain 8, 15-bit general purpose registers R0 – R7. A 15-bit program counter PC, 15-bit instruction buffer IB, 15-bit save restore program counter, an 8-bit flags register, and an 8-bit save restore flags register. The flags register has two empty bits, and six flags: Always A, Interrupt I, Zero Z, Negative N, Overflow O, Carry C, Interrupt Enable E. All instructions are 15-bits and are described in the following section.

CPU Inputs:

- CLK – The CPU Clock
- IRQ – Interrupt request
- DATAIN – The 15-bit data path in from memory and I/O

CPU Outputs:

- ADDR – The 16-bit memory address to be read or written
- RE – The read enable to memory, high when data is to be read from memory
- WE – The write enable to memory, high when data is to be written to memory
- DATAOUT – The 15-bit data path out to memory and I/O

CPU Debug Outputs:

- PC – The 15-bit program counter
- IB – The 15-bit instruction buffer
- SRP – The 15-bit save restore program counter
- SRF – The 8-bit save restore flags register
- R0 – R7 – The 8, 15-bit general purpose registers
- I – Interrupt flag
- Z – Zero flag
- N – Negative flag
- O – Overflow flag
- C – Carry flag
- E – Interrupt Enable flag

The mapping of the flags to flag numbers, and the instructions to instruction number I4..I0 is left up to each group. You need to describe your rationale for choosing your instruction mapping. Your CPU will be connected to a small memory that will be used for both data and instructions. You will be provided several small assembly programs, an assembler, and a given test circuit in order to test your CPU. The adder for your ALU must be implemented using multiple 5-bit carry look-ahead units, or built as a prefix adder.

When an interrupt occurs (either IRQ with E flag set, or SWI instruction), during the writeback stage:

- The address 0x7FFF from memory updates the PC
- The E flag is cleared
- The SRP is updated with PC for IRQ interrupt or PC + 1 for SWI
- The SRF is updated with the flags register

An assembler has been provided as a Python 3 script that takes in a CSV that specifies the mapping of flags and instructions to machine code. There are several assembly programs that have been provided for you to test. There is an empty translation CSV file that needs to have flag indices between 0 – 7 set, and the bit pattern for each instruction specified. A Python 3 script that will convert a string to a set of DAT commands has been provided as well.

An approach you may want to take for this project is as follows:

1. Implement the 15-bit adder by either starting with the 5-bit carry look-ahead or doing a full prefix adder.
2. Implement the 15-bit ALU after deciding on instruction mapping, and using the adder from step 1. Keep in mind that the ALU will likely need to calculate some of the flags.
3. Implement register file. This should contain at least:
    a. A clock signal input
    b. 8, 15-bit general purpose registers
    c. A selector for write target register
    d. An enable for the write update
    e. An 15-bit input for the write value
    f. Two selectors for the register outputs
    g. Two 15-bit output ports
4. Create a CPU that will read an instruction into the IB based upon the PC, and then implements NOP for the writeback stage. The PC should be incremented in the writeback stage so that the next memory address will be read.
5. Add the ALU and register file to allow for register/register instructions, and also for loading immediate values.
6. Expand CPU to implement the branching and jump instructions.
7. Implement the interrupt, SWI, and RTI instructions.

LIL0 (Load Immediate Low 0)

Description

Load Immediate Low loads a 7-bit immediate value into the least significant byte of the target register with 0 in the most significant bit of the least significant byte.

LIL0 T, #V

0 V_6-V_0 → RT_7-RT_0

PC + 1 → PC

Flags Effected: None

Opcode

Bit	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	I_4	I_3	I_2	I_1	I_0	T_2	T_1	T_0	V_6	V_5	V_4	V_3	V_2	V_1	V_0

LIM1 (Load Immediate Low 1)

Description

Load Immediate Medium loads a 7-bit immediate value into the least significant byte of the target register with 1 in the most significant bit of the least significant byte.

LIL1 T, #V

1 V_6-V_0 → RT_7-RT_0

PC + 1 → PC

Flags Effected: None

Opcode

Bit	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	I_4	I_3	I_2	I_1	I_0	T_2	T_1	T_0	V_6	V_5	V_4	V_3	V_2	V_1	V_0

LIH (Load Immediate High)

Description

Load Immediate High loads a 7-bit immediate value into the most significant 7 bits of the target register.

LIH T, #V

V_6-V_0 → RT_{14}-RT_8

PC + 1 → PC

Flags Effected: None

Opcode

Bit	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	I_4	I_3	I_2	I_1	I_0	T_2	T_1	T_0	V_6	V_5	V_4	V_3	V_2	V_1	V_0

LD (Load)

Description

Load, loads data from main memory into a register. The address of memory is specified by the address register A. A four bit signed offset O is added to the address of the source.

LD T, A + O

Mem(A + O) → T

PC + 1 → PC

Flags Effected: None

Opcode

Bit	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	I_4	I_3	I_2	I_1	I_0	T_2	T_1	T_0	A_2	A_1	A_0	O_3	O_2	O_1	O_0

ST (Store)

Description

Store, stores data from a register to main memory. The address of memory is specified by the address register A. A four bit signed offset O is added to the address of the source.

ST S, A + O

S → Mem(A + O)

PC + 1 → PC

Flags Effected: None

Opcode

Bit	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	I_4	I_3	I_2	I_1	I_0	S_2	S_1	S_0	A_2	A_1	A_0	O_3	O_2	O_1	O_0

ADD (Add)

Description

Add, adds two registers together and stores the value into a destination register.

ADD C, A, B

A + B → C

PC + 1 → PC

Flags Effected: Z, N, O, and C.

Opcode

Bit	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	I_4	I_3	I_2	I_1	I_0	C_2	C_1	C_0	A_2	A_1	A_0	B_2	B_1	B_0	X

SUB (Subtract)

Description

Subtract, subtracts register from another and stores the value into a destination register.

SUB C, A, B

A - B → C

PC + 1 → PC

Flags Effected: Z, N, O, and C.

Opcode

Bit	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	I_4	I_3	I_2	I_1	I_0	C_2	C_1	C_0	A_2	A_1	A_0	B_2	B_1	B_0	X

ADDI (Add Immediate)

Description

Add Immediate, adds an immediate 4-bit signed value to a register and stores it back into a destination register.

ADD C, A, #V

A + V → C

PC + 1 → PC

Flags Effected: Z, N, O, and C.

Opcode

Bit	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	I_4	I_3	I_2	I_1	I_0	C_2	C_1	C_0	A_2	A_1	A_0	V_3	V_2	V_1	V_0

AND (And)

Description

And, ands two registers together and stores the resulting value back into a destination register.

AND C, A, B

A & B → C

PC + 1 → PC

Flags Effected: Z, and N.

Opcode

Bit	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	I_4	I_3	I_2	I_1	I_0	C_2	C_1	C_0	A_2	A_1	A_0	B_2	B_1	B_0	X

OR (Or)

Description

Or, ors two registers together and stores the resulting value back into a destination register.

OR C, A, B

A | B → C

PC + 1 → PC

Flags Effected: Z, and N.

Opcode

Bit	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	I_4	I_3	I_2	I_1	I_0	C_2	C_1	C_0	A_2	A_1	A_0	B_2	B_1	B_0	X

XOR (Or)

Description

Xor, xors two registers together and stores the resulting value back into a destination register.

XOR C, A, B

A ^ B → C

PC + 1 → PC

Flags Effected: Z, and N.

Opcode

Bit	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	I_4	I_3	I_2	I_1	I_0	C_2	C_1	C_0	A_2	A_1	A_0	B_2	B_1	B_0	X

NOT (Not)

Description

Not, flips all the bits of the source register and stores the value back into the destination register.

NOT C, A

$\bar{A}$ → C

PC + 1 → PC

Flags Effected: Z, and N.

Opcode

Bit	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	I_4	I_3	I_2	I_1	I_0	C_2	C_1	C_0	A_2	A_1	A_0	X	X	X	X

INV (Invert)

Description

Invert, negates the two's complement value of the source register and stores the value back into the destination register.

INV C, A

-A → C

PC + 1 → PC

Flags Effected: Z, and N.

Opcode

Bit	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	I_4	I_3	I_2	I_1	I_0	C_2	C_1	C_0	A_2	A_1	A_0	X	X	X	X

MOV (Move)

Description

Move, moves the value of the source register into the destination register.

MOV C, A

A → C

PC + 1 → PC

Flags Effected: None

Opcode

Bit	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	I_4	I_3	I_2	I_1	I_0	C_2	C_1	C_0	A_2	A_1	A_0	X	X	X	X

ROR (Rotate Right)

Description

Rotate right, rotates the bits of the source register right and stores the value into the destination register. The carry bit is rotated into the most significant bit, and the least significant bit is rotated into the carry bit.

ROR C, A

$A_N \rightarrow C_{N-1}$

PC + 1 → PC

Flags Effected: Z, N, C.

Opcode

Bit	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	I_4	I_3	I_2	I_1	I_0	C_2	C_1	C_0	A_2	A_1	A_0	X	X	X	X

ROL (Rotate Left)

Description

Rotate left, rotates the bits of the source register left and stores the value into the destination register. The carry bit is rotated into the least significant bit, and the most significant bit is rotated into the carry bit.

ROL C, A

$A_N \rightarrow C_{N+1}$

PC + 1 → PC

Flags Effected: Z, N, C.

Opcode

Bit	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	I_4	I_3	I_2	I_1	I_0	C_2	C_1	C_0	A_2	A_1	A_0	X	X	X	X

SHR (Shift Right)

Description

Shift right, shifts the bits of the source register right and stores the value into the destination register. The most significant bit is maintained, and the least significant bit is rotated into the carry bit.

SHR C, A

$A_N \rightarrow C_{N-1}$

PC + 1 → PC

Flags Effected: Z, N, C.

Opcode

Bit	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	I_4	I_3	I_2	I_1	I_0	C_2	C_1	C_0	A_2	A_1	A_0	X	X	X	X

SHL (Shift Left)

Description

Shift left, rotates the bits of the source register left and stores the value into the destination register. The least significant bit is set to zero, and the most significant bit is rotated into the carry bit.

SHL C, A

$A_N \rightarrow C_{N+1}$

PC + 1 → PC

Flags Effected: Z, N, C.

Opcode

Bit	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	I_4	I_3	I_2	I_1	I_0	C_2	C_1	C_0	A_2	A_1	A_0	X	X	X	X

BR (Branch)

Description

Branch, branches the Program Counter PC to the relative value of the offset if the flag is set that is specified by the flag bits. Otherwise the Program Counter is incremented by one.

BR F, #O

IF F-bit set PC + O → PC

ELSE PC + 1 → PC

Flags Effected: None.

Opcode

Bit	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	I_4	I_3	I_2	I_1	I_0	F_2	F_1	F_0	O_6	O_5	O_4	O_3	O_2	O_1	O_0

JMP (Jump)

Description

Jump sets the Program Counter to the value specified in the register.

JMP A

A → PC

Flags Effected: None.

Opcode

Bit	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	I_4	I_3	I_2	I_1	I_0	A_2	A_1	A_0	X	X	X	X	X	X	X

JSR (Jump Subroutine)

Description

Jump Subroutine sets the Program Counter to the value specified in the register and stores the previous PC + 1 in the destination register.

JSR C, A

A → PC

PC + 1 → C

Flags Effected: None.

Opcode

Bit	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	I_4	I_3	I_2	I_1	I_0	C_2	C_1	C_0	A_2	A_1	A_0	X	X	X	X

NOP (No Operation)

Description

NOP does no operation but increments the Program Counter.

NOP

PC + 1 → PC

Flags Effected: None.

Opcode

Bit	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	I_4	I_3	I_2	I_1	I_0	X	X	X	X	X	X	X	X	X	X

LDFI (Load Flag Immediate)

Description

Load Flag Immediate loads an immediate value into a specific flag in the flag register.

LDFI A, #V

$V \rightarrow F_A$

PC + 1 → PC

Flags Effected: F_A

Opcode

Bit	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	I_4	I_3	I_2	I_1	I_0	A_2	A_1	A_0	X	X	X	X	X	X	V

MOVF (Move Flags)

Description

Move flags, moves either the low byte of the source register into the flags register, or moves the flags register into the low byte of the destination register.

MOVF A, #1 or MOVF B, #0

IF D set $F \rightarrow A_7\text{-}A_0$

ELSE $B_7\text{-}B_0 \rightarrow F$

PC + 1 → PC

Flags Effected: None.

Opcode

Bit	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	I_4	I_3	I_2	I_1	I_0	A_2	A_1	A_0	B_2	B_1	B_0	X	X	X	D

MSRP (Move Save Restore PC)

Description

Move save restore PC, moves either the source register into the SRP, or moves the SRP register into the destination register.

MSRP A, #1 or MSRP B, #0

IF D set MSRP → A

ELSE B → MSRP

PC + 1 → PC

Flags Effected: None.

Opcode

Bit	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	I_4	I_3	I_2	I_1	I_0	A_2	A_1	A_0	B_2	B_1	B_0	X	X	X	D

MSRF (Move Save Restore Flags)

Description

Move save restore flags, moves either the low byte of the source register into the SRF register, or moves the SRF register into the low byte of the destination register.

MSRF A, #1 or MSRF B, #0

IF D set SRF → A_7-A_0

ELSE B_7-B_0 → SRF

PC + 1 → PC

Flags Effected: None.

Opcode

Bit	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	I_4	I_3	I_2	I_1	I_0	A_2	A_1	A_0	B_2	B_1	B_0	X	X	X	D

SWI (Software Interrupt)

Description

SWI generates a software interrupt that will be processed regardless of the E flag value. The interrupt vector is stored in memory location 0x7FFF (maximum address) and is fetched from the memory location to be loaded into the PC. The PC + 1 (address of the instruction after SWI) is stored in the SRP, the current flags register is stored in SRF, and the E flag is cleared.

SWI

Mem(0x7FFF) → PC

PC + 1 → SRP

F → SRF

Flags Effected: E.

Opcode

Bit	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	I_4	I_3	I_2	I_1	I_0	X	X	X	X	X	X	X	X	X	X

RTI (Return From Interrupt)

Description

RTI returns from an interrupt. The PC is updated with the value of the SRP, and the flags are updated with the value of SRF.

RTI

SRP → PC

SRF → F

Flags Effected: All.

Opcode

Bit	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	I_4	I_3	I_2	I_1	I_0	X	X	X	X	X	X	X	X	X	X