BLG222E Computer Organization

Project 2

Due Date: 09.05.2024, 23:59

Design a **hardwired control unit** for the following architecture. Use the structure that you have designed in **Part 4 of Project 1.**

INSTRUCTION FORMAT

The instructions are stored in memory in **little-endian order.** Since the RAM in Project 1 has an 8-bit output, **the instruction register cannot be filled in one clock cycle.** You can load MSB and LSB in 2 clock cycles.

- In the first clock cycle (**T=0**), the LSB of the instruction must be loaded from an address A of the memory to the LSB of IR [i.e., IR(7-0)].
- In the second clock cycle (**T=1**), the MSB of the instruction must be loaded from an address A+1 of the memory to the MSB of IR [i.e., IR(15-8)].
- After the second clock cycle (**T=2**), the instruction starts to execute.

There are 2 types of instructions as described below.

1- Instructions with address reference have the format shown in Figure 1.

- The **OPCODE** is a **6-bit field.** (Table 1 is given for opcode definition.)
- The **RSEL** is a **2-bit field.** (Table 2 is given for register selection.)
- The **ADDRESS** is an **8-bit field.**

OPCODE (6-bit)	RSEL (2-bit)	ADDRESS (8-bit)

Figure 1: Instructions with an address reference.

2- Instructions without an address reference have the format shown in Figure 2.

- The **OPCODE** is a **6-bit field.** (Table 1 is given for opcode definition.)
- The **S** is a **1-bit field** that specifies whether the flags will change or not.
- The **DSTREG** is a **3-bit field** that specifies the destination register. (Table 3 is given.)
- The **SREG1** is a **3-bit field** that specifies the first source register. (Table 3 is given.)
- The **SREG2** is a **3-bit field** that specifies the second source register. (Table 3 is given.)

OPCODE (6-bit)	S (1-bit)	DSTREG(3-bit)	SREG1 (3-bit)	SREG2 (3-bit)

Figure 2: Instructions without an address reference.

Table 1: OPCODE field and symbols for operations and their descriptions.

OPCODE (HEX)	SYMBOL	DESCRIPTION
0x00	BRA	PC ← PC + VALUE
0x01	BNE	IF Z=0 THEN PC ← PC + VALUE
0x02	BEQ	IF Z=1 THEN PC ← PC + VALUE
0x03	POP	SP ← SP + 1, Rx ← M[SP]
0x04	PSH	M[SP] ← Rx, SP ← SP – 1
0x05	INC	DSTREG ← SREG1 + 1
0x06	DEC	DSTREG ← SREG1 – 1
0x07	LSL	DSTREG ← LSL SREG1
0x08	LSR	DSTREG ← LSR SREG1
0x09	ASR	DSTREG ← ASR SREG1
0x0A	CSL	DSTREG ← CSL SREG1
0x0B	CSR	DSTREG ← CSR SREG1
0x0C	AND	DSTREG ← SREG1 AND SREG2
0x0D	ORR	DSTREG ← SREG1 OR SREG2
0x0E	NOT	DSTREG ← NOT SREG1
0x0F	XOR	DSTREG ← SREG1 XOR SREG2
0x10	NAND	DSTREG ← SREG1 NAND SREG2
0x11	MOVH	DSTREG[15:8] ← IMMEDIATE **(8-bit)**
0x12	LDR (16-bit)	Rx ← M[AR] (**AR is 16-bit register**)
0x13	STR (16-bit)	M[AR] ← Rx (**AR is 16-bit register**)
0x14	MOVL	DSTREG[7:0] ← IMMEDIATE **(8-bit)**
0x15	ADD	DSTREG ← SREG1 + SREG2
0x16	ADC	DSTREG ← SREG1 + SREG2 + CARRY
0x17	SUB	DSTREG ← SREG1 - SREG2
0x18	MOVS	DSTREG ← SREG1, Flags will change
0x19	ADDS	DSTREG ← SREG1 + SREG2, Flags will change
0x1A	SUBS	DSTREG ← SREG1 - SREG2, Flags will change
0x1B	ANDS	DSTREG ← SREG1 AND SREG2, Flags will change
0x1C	ORRS	DSTREG ← SREG1 OR SREG2, Flags will change
0x1D	XORS	DSTREG ← SREG1 XOR SREG2, Flags will change
0x1E	BX	M[SP] ← PC, PC ← Rx
0x1F	BL	PC ← M[SP]
0x20	LDRIM	Rx ← VALUE (VALUE defined in ADDRESS bits)
0x21	STRIM	M[AR+OFFSET] ← Rx (**AR is 16-bit register**) (OFFSET defined in ADDRESS bits)

Table 2: RSel table.

RSEL	REGISTER
00	R1
01	R2
10	R3
11	R4

Table 3: DSTREG/SREG1/SREG2 selection table.

DSTREG/SREG1/SREG2	REGISTER
000	PC
001	PC
010	SP
011	AR
100	R1
101	R2
110	R3
111	R4

SIMPLE EXAMPLE

Since the PC value is initially 0, your code first executes the instruction in memory address 0x00. This instruction is BRA START_ADDRESS where START_ADDRESS is the starting address of your instructions.

You have to determine the binary code of the program and write it to memory. Your final Verilog implementation is expected to fetch the instructions starting from address 0x00, decode them, and execute all instructions one by one.

```
                BRA 0x28            # This instruction is written to the memory address 0x00,
                                    # The first instruction must be written to address 0x28
                MOVH R1, 0x00
                MOVL R1, 0x0A       # This first instruction is written to the address 0x28,
                                    # R1 is used for the number of iterations
                MOVH R2, 0x00
                MOVL R2, 0x00       # R2 is used to store total
                MOVH R3, 0x00
                MOVL R3, 0xB0
                MOVS AR, R3         # AR is used to track data address: starts from 0xA0
LABEL:          LDR R3              # R3 ← M[AR] (AR = 0xB0 to 0xBA)
                ADD R2, R2, R3      # R2 ← R2 + R3 (Total = Total + M[AR])
                INC AR, AR          # AR ← AR + 1 (Next Data)
                DEC R1, R1          # R1 ← R1 – 1 (Decrement Iteration Counter)
                BNE LABEL           # Go back to LABEL if Z=0 (Iteration Counter > 0)
                INC AR, AR          # AR ← AR + 1 (Total will be written to 0xBB)
                STR R2              # M[AR] ← R2 (Store Total at 0xBB)
```

Submission: Implement your design in Verilog HDL, and upload your design, simulation & Memory files to Ninova before the deadline. Only one student from each group should submit the project files (select one member of the group as the group representative for this purpose and note his/her student ID). Project files should contain your modules file (.v), simulation file (.v), memory (.mem), and a report that contains:

- the number & names of the students in the group
- information about your control unit design

If you are working as a group, all members of the group must design together. Make sure to add simulations for all modules. You must ensure that all modules work properly.