BLG222E Computer Organization
Project 2
Due Date: 24.05.2023, 23:00

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

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 **4-bit field**. (Table 1 is given for opcode definition)
- The next bit (**IR[11]**) is unused, it is set as 0.
- The **ADDRESSING MODE** is a **1-bit field.** (Table 2 is given for addressing.)
- The **RSEL** is a **2-bit field.** (Table 3 is given for register selection.)
- The **ADDRESS** is an **8-bit field.**

OPCODE (4-bit)	0 (1-bit)	ADDRESSING MODE (1-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 Fİgure 2.

- The **OPCODE** is a **4-bit field**. (Table 1 is given for opcode definition)
- The **DSTREG** is a **4-bit field** which specifies the destination register. (Table 4 is given.)
- The **SREG1** is a **4-bit field** which specifies the first source register. (Table 4 is given.)
- The **SREG2** is a **4-bit field** which specifies the second source register. (Table 4 is given.)

OPCODE (4-bit)	DSTREG (4-bit)	SREG1 (4-bit)	SREG2 (4-bit)

Figure 2: Instructions without an address reference.

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

OPCODE (HEX)	SYMBOL	ADDRESSING MODE	DESCRIPTION
0x00	AND	N/A	DSTREG←SREG1 AND SREG2
0x01	OR	N/A	DSTREG←SREG1 OR SREG2
0x02	NOT	N/A	DSTREG←NOT SREG1
0x03	ADD	N/A	DSTREG←SREG1+SREG2
0x04	SUB	N/A	DSTREG←SREG1-SREG2
0x05	LSR	N/A	DSTREG←LSR SREG1
0x06	LSL	N/A	DSTREG←LSL SREG1
0x07	INC	N/A	DSTREG←SREG1+1
0x08	DEC	N/A	DSTREG←SREG1-1
0x09	BRA	IM	PC←VALUE
0x0A	BNE	IM	IF Z=0 THEN PC←VALUE
0x0B	MOV	N/A	DSTREG←SREG1
0x0C	LD	IM, D	Rx←VALUE (described in Table-3)
0x0D	ST	D	VALUE←Rx
0x0E	PUL	N/A	SP ← SP + 1, Rx ← M[SP]
0x0F	PSH	N/A	M[SP] ← Rx, SP ← SP - 1

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
0000	R1
0001	R2
0010	R3
0011	R4
0100	SP
0101	AR
0110	PC
0111	PC

Table-4: Addressing modes.

ADDRESSING MODE	MODE	SYMBOL	VALUE
0	Immediate	IM	ADDRESS field
1	Direct	D	M[AR]

SIMPLE LOOP EXAMPLE

Since 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 the 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 the address 0x28
        LD R1 IM 0x0A     # This first instruction is written to the address 0x28,
                          # R1 is used for iteration number
        LD R2 IM 0x00     # R2 is used to store total
        LD R3 IM 0xB0
        MOV AR R3         # AR is used to track data address: starts from 0xA0
LABEL:  LD R3 D           # 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 IM LABEL      # Go back to LABEL if Z=0 (Iteration Counter > 0)
        INC AR AR          # AR ← AR + 1 (Total will be written to 0xBB)
        ST R2 D           # M[AR] ← R2 (Store Total at 0xBB)
```

Note: Testbench code will be shared later.

Submission: Implement your design in Verilog HDL, upload your design, simulation, and 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

Group work is expected for this project. All the 3 student members of the group must design together. Make sure to add simulations for all modules. You must ensure that all modules work properly. Simulation files will be altered in the demonstration session. Please do not hesitate to contact assistants for any question.