BLG222E Computer Organization

Project 2

Due Date: 1 JUNE 2022, 18:00

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, **IR** cannot be filled in one clock cycle. Fortunately, you have **L/H** signal in the **IR** and you can load MSB and LSB separately in two 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 two 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 (See Table 1 for the definition).
- The next bit [i.e., **IR(11)**] is unused, you set it as 0
- The **ADDRESSING MODE** is a **1-bit** field (See Table 3 for the definition).
- The **REGSEL** is a **2-bit** field (See left side of Table 2 for the definition).
- The **ADDRESS** is an **8-bit** field.

OPCODE (4-bit)	0 (1-bit)	ADDRESSING MODE (1-bit)	REGSEL (2-bit)	ADDRESS (8-bit)

Figure 1: Instructions with an address reference

(2) Instructions without address reference have the format shown in Figure 2:

- The **OPCODE** is a **4-bit** field (See Table 1 for the definition).
- The **DESTREG** is a **4-bit** field which specifies the destination register (See right side of Table 2 for the definition).
- The **SRCREG1** is a **4-bit** field which specifies the first source register (See right side of Table 2 for the definition).
- The **SRCREG2** is a **4-bit** field which specifies the second source register (See right side of Table 2 for the definition).

OPCODE (4-bit)	DESTREG (4-bit)	SRCREG1 (4-bit)	SRCREG2 (4-bit)

Figure 2: Instructions without an address reference

Table 1: OPCODE field and SYMBols for operations and their descriptions

OPCODE (HEX)	SYMB	ADDRESSING MODE	DESCRIPTION
0x00	BRA	IM	PC ← Value
0x01	LD	IM, D	Rx ← Value (Value is described in Table 3)
0x02	ST	D	Value ← Rx
0x03	MOV	N/A	DESTREG ← SRCREG1
0x04	AND	N/A	DESTREG ← SRCREG1 AND SRCREG2
0x05	OR	N/A	DESTREG ← SRCREG1 OR SRCREG2
0x06	NOT	N/A	DESTREG ← NOT SRCREG1
0x07	ADD	N/A	DESTREG ← SRCREG1 + SRCREG2
0x08	SUB	N/A	DESTREG ← SRCREG2 - SRCREG1
0x09	LSR	N/A	DESTREG ← LSR SRCREG1
0x0A	LSL	N/A	DESTREG ← LSL SRCREG1
0x0B	PUL	N/A	SP <- SP + 1, Rx <- M[SP]
0x0C	PSH	N/A	M[SP] <- Rx, SP <- SP - 1
0x0D	INC	N/A	DESTREG ← SRCREG1 + 1
0x0E	DEC	N/A	DESTREG ← SRCREG1 - 1
0x0F	BNE	IM	IF Z=0 THEN PC ← Value

Table 2:REGSEL (Left) and DESTREG/SRCREG1/SRCREG2 (Right) select the register of interest for a particular instruction

REGSEL	REGISTER
00	R1
01	R2
10	R3
11	R4

DESTREG/SRCREG1/SRCREG2	REGISTER
0000	PC
0001	PC
0010	AR
0011	SP
0100	R1
0101	R2
0110	R3
0111	R4

Note: SRCREG1 will be used for all registers in Table 2, SRCREG2 will be used for R1, R2, R3, and R4.

Table 3: Addressing modes

ADDRESSING MODE	MODE	SYMB	Value
0	Direct	D	M[AR]
1	Immediate	IM	ADDRESS Field

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.

The code given below adds data that are stored at memory addresses 0xA0, 0xA1, 0xA2, 0xA3, and 0xA4 (i.e., calculates **total** = M[A0] + M[A1] + M[A2] + M[A3] + M[A4]). Then, it stores the **total** in memory address 0xA6 (i.e., M[A6]). It is written as a loop that iterates 5 times.

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  0x20          # This instruction is written to the memory address 0x00,
                           # The first instruction must be written to the address 0x20


        LD R1 IM 0x05      # This first instruction is written to the address 0x20,
                           # R1 is used for iteration number
        LD R2 IM 0x00      # R2 is used to store total
        LD R3 IM 0xA0
        MOV AR R3          # AR is used to track data address: starts from 0xA0
LABEL:  LD R3 D            # R3 ← M[AR]        (AR = 0xA0 to 0xA4)
        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 0xA6)
        ST  R2 D           # M[AR] ← R2        (Store Total at 0xA6)
```

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 Res. Asst. Kadir Özlem (kadir.ozlem@itu.edu.tr) for any question.