ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 222E
COMPUTER ORGANIZATION PROJECT 2 REPORT

PROJECT NO	:	2
PROJECT DATE	:	24.05.2023
GROUP NO	:	G24

GROUP MEMBERS:

150200029	:	HAVVA EDA KÖRPE
150200066	:	YUSUF EMİR SEZGİN
150200033	:	ÖZKAN GEZMİŞ

SPRING 2023

Contents

1 INTRODUCTION **1**

2 MATERIALS AND METHODS **1**

2.1 FETCH AND DECODE 1

2.2 OPERATIONS 1

2.2.1 **AND** Operation 1

2.2.2 **OR** Operation 1

2.2.3 **NOT** Operation 2

2.2.4 **ADD** Operation 2

2.2.5 **SUB** Operation 2

2.2.6 **LSR** Operation 2

2.2.7 **LSL** Operation 2

2.2.8 **INC** Operation 2

2.2.9 **DEC** Operation 2

2.2.10 **BRA** Operation 3

2.2.11 **BNE** Operation 3

2.2.12 **MOV** Operation 3

2.2.13 **LD** Operation 3

2.2.14 **ST** Operation 3

2.2.15 **PUL** Operation 3

2.2.16 **PSH** Operation 3

2.3 MEMORY 4

3 RESULTS **4**

4 DISCUSSION **5**

5 CONCLUSION **5**

1 INTRODUCTION

Data operations that are done with registers are called microoperations. To obtain the results of these operations Arithmetic Logic Units (ALU) are used in the CPU. Hardwired Control Units (HCU) are used to operate these microoperations using ALU and Memory. We designed an HCU that is connected with ALU to operate 16 different operations. The HCU uses ALU's outputs as inputs and runs accordingly to given operations. HCU decides what operation needs to be operated by looking at memory addresses. Finally, we implemented an example run to clarify our design.

2 MATERIALS AND METHODS

Providing the ALU System that we used in our previous project, we have designed a new system that takes instructions from memory and controls the process of different operations.

2.1 FETCH AND DECODE

Initially, we start with setting our inputs which are RF FunSel and ARF FunSel to 00 which clears the register. And RF RegSel and ARF RegSel to 1111 which enables all the registers. Then, the Instruction register is enabled and cleared with proper inputs. We will increment Sequence Counter in each clock cycle. At t0 time, we are loading (0-7) bits of the instruction from the memory reading the address from the memory. At time t1, (8-16) bits are also loaded from the memory to IR. Then, at time t2, regarding (15-12) bits of IR, we are decoding the operations with conditional blocks as in the given table to us.

2.2 OPERATIONS

2.2.1 AND Operation

The AND operation takes two registers which are selected with SREG1 and SREG2, performs a **boolean AND operation** for the values of these two, then copies the changed value to the destination register which is selected with DSREG.

2.2.2 OR Operation

The OR operation takes two registers which are selected with SREG1 and SREG2, performs a **boolean OR operation** for these two, then copies the changed value to the destination register which is selected with DSREG.

2.2.3 NOT Operation

The NOT operation takes the register that is selected with SREG1, and **complements** its value, then copies the changed value to the destination register which is selected with DSREG.

2.2.4 ADD Operation

The ADD operation takes two registers which are selected with SREG1 and SREG2, performs an **addition** between their values, then copies the changed value to the destination register which is selected with DSREG.

2.2.5 SUB Operation

The SUB operation takes two registers which are selected with SREG1 and SREG2, performs a **subtraction** between their values, then copies the changed value to the destination register which is selected with DSREG.

2.2.6 LSR Operation

The LSR operation takes the register that is selected with SREG1, **right shifts its value (binary shift)**, then copies the changed value to the destination register which is selected with DSREG.

2.2.7 LSL Operation

The LSL operation takes the register that is selected with SREG1, **left shifts its value (binary shift)**, then copies the changed value to the destination register which is selected with DSREG.

2.2.8 INC Operation

The INC operation takes the register that is selected with SREG1, **increments its value by one**, then copies the changed value to the destination register which is selected with DSREG.

2.2.9 DEC Operation

The DEC operation takes the register that is selected with SREG1, **decrements its value by one**, then copies the changed value to the destination register which is selected with DSREG.

2.2.10 BRA Operation

In the BRA operation, we use the instruction which contains an 8-bit address. In this operation, we use an immediate address which means taking the address directly as value and transferring it to the PC.

2.2.11 BNE Operation

In the BNE operation, we use the instruction which contains an 8-bit address. In this operation, we control the flag Z and we use an immediate address which means taking the address directly as value and transferring it to the PC.

2.2.12 MOV Operation

The MOV operation takes the register that is selected with SREG1 and copies **(moves)** the value to the destination register which is selected with DSREG.

2.2.13 LD Operation

The LD operation is a loading operation that takes the value and transfers it to the chosen register which is selected by RSEL. The addressing mode bit in the instruction code determines the value to load.

2.2.14 ST Operation

The ST operation is a storing operation that takes the value of the register that is chosen and writes it to the given address with a value. To specify the address where the value will be stored is specified considering the addressing mode bit of the instruction.

2.2.15 PUL Operation

The PUL operation is a stack operation that increments the value of the sequence pointer. And it takes the value in the address of the sequence pointer and copies it into the selected register.

2.2.16 PSH Operation

The PSH operation is a stack operation that takes the value of the selected register and copies it into the address which is specified by the sequence pointer. Then, it decrements the value of the sequence pointer.

2.3 MEMORY

We have inserted instructions to the memory determining and merging the different parts of it. We replaced LSB in the first step, MSB in the second step. Operations in the memory is as follows:

01 0x00: BRA 0x28 = 90 28
41 0x28: LD R1 IM 0x0A = C0 0A
43 0x2A: LD R2 IM 0x00 = C1 00
45 0x2C: LD R3 IM 0xB0 = C2 B0
47 0x2E: MOV AR R3 = B5 20
49 0x30: LABEL: LD R3 D = C6 00
51 0x32: ADD R2 R2 R3 = 31 12
53 0x34: INC AR AR = 75 50
55 0x36: DEC R1 R1 = 80 00
57 0x38: BNE IM LABEL = A0 30
59 0x3A: INC AR AR = 75 50
61 0x3C: ST R2 D = D5 00

3 RESULTS

We have experienced that, by using the HCU system connected with the ALU system, we can obtain register operations that can perform different types of tasks.

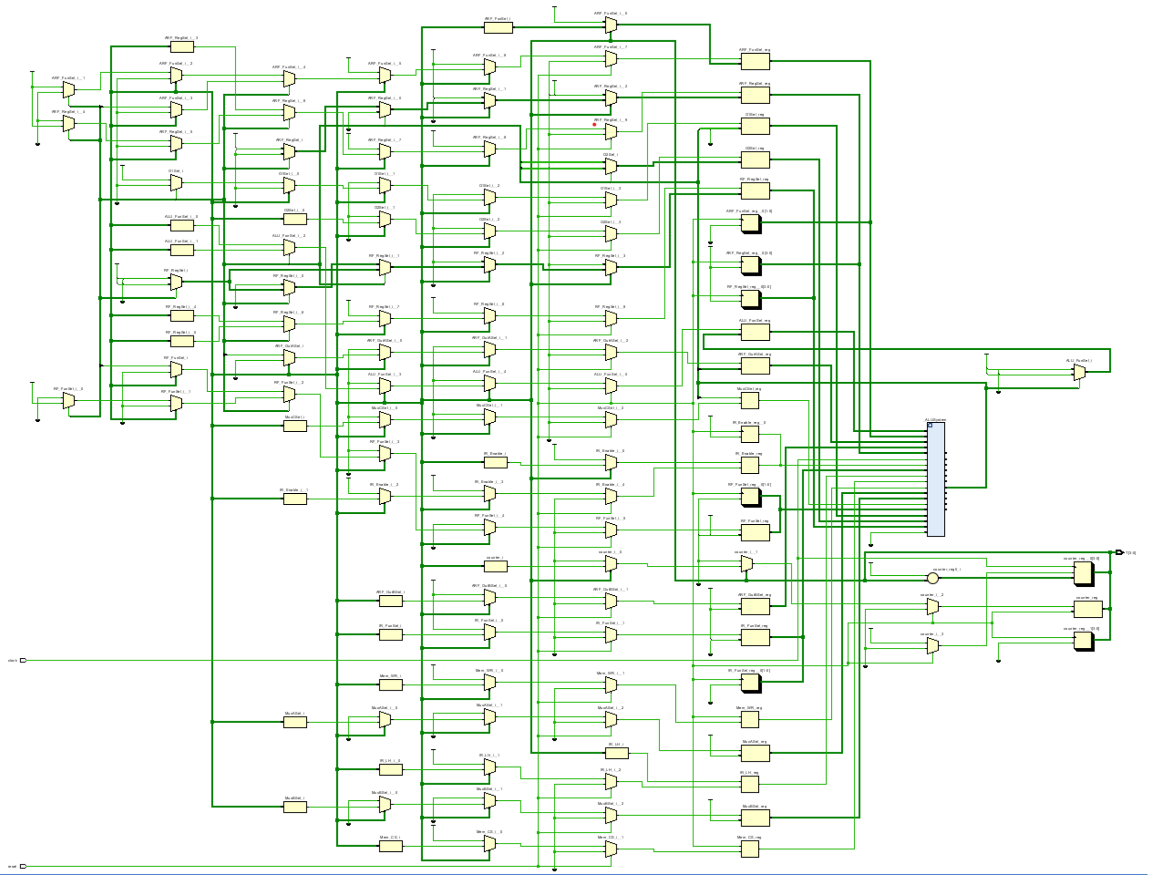

Figure 1: Instructions in the Memory

4 DISCUSSION

Firstly, we designed the clock which starts from t0 and counts to t16. We have used this clock to manage different operation types. We have two different instruction types. One of them is for operations that require an address for the memory, the other one is for selecting two registers to operate on their values. In each conditional block, we have decided which one of them to use and also considered the addressing mode, if it is necessary, to perform these operations. With these processes, we have added a hardware control unit to our previous system that contains ALU and prepared inputs according to that considering register and function selections of this system. Finally, to perform the given operations, there is a need for memory that contains instructions.

5 CONCLUSION

In this project, we have created a hardware control unit using appropriate clock signals. In each different operation, register selections and address types change. AND, OR, NOT, ADD, SUB, LSR, LSL, INC, DEC, BRA, BNE, MOV, LD, ST, PUL, and PSH operations has been performed with different opcode values.