ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 222E

COMPUTER ORGANIZATION
FINAL PROJECT REPORT

PROJECT NO	:	4
PROJECT DATE	:	14.07.2020
GROUP NO	:	36

GROUP MEMBERS:

150170087 : Sırrı Batuhan ÇOKSAK(Group Representative)

150170069 : Furkan Yusuf GÜRAY

150170039 : Fatih MURAT

150160142 : Ekin Zuhat YAŞAR

SPRING 2020

Contents

1 INTRODUCTION

In the final project, we were expected to design and implement software-based (microprogrammed) control unit for the architecture in the Figure-1.

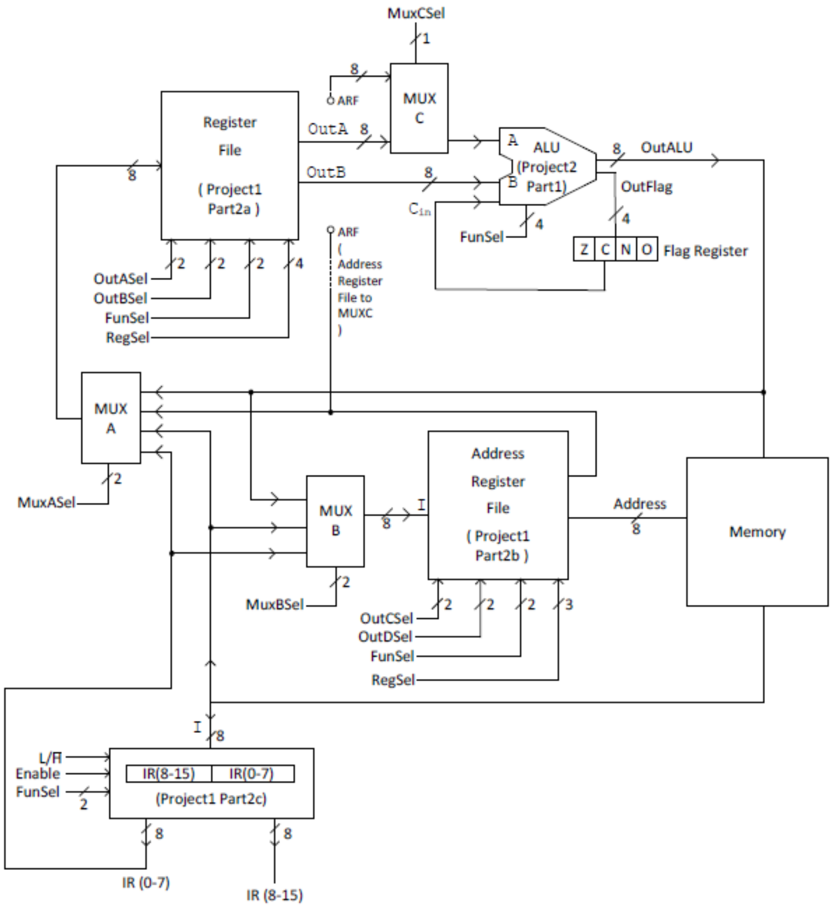

Figure 1: Template architecture

2 DESIGN

In the project we were given 19 different opcodes in the Figure2, we analyzed the opcodes and decided our mapping format which would be "1 + opcode + 00" which means that we will be using 4 line of micro-instructions to perform the operation of the opcode, the added 1 bit to the start of the mapping will aid some issues which will be discussed in discussion part of the report.

OPCODE (HEX)	SYMB	ADDRESSING MODE	DESCRIPTION
0x00	LD	IM, D	Rx ← Value (Value is described in Table 3)
0x01	ST	D	Value ← Rx
0x02	MOV	N/A	DESTREG ← SRCREG1
0x03	PSH	N/A	M[SP] ← Rx, SP ← SP - 1
0x04	PUL	N/A	SP ← SP + 1, Rx ← M[SP]
0x05	ADD	N/A	DESTREG ← SRCREG1 + SRCREG2
0x06	SUB	N/A	DESTREG ← SRCREG2 - SRCREG1
0x07	DEC	N/A	DESTREG ← SRCREG1 - 1
0x08	INC	N/A	DESTREG ← SRCREG1 + 1
0x09	AND	N/A	DESTREG ← SRCREG1 AND SRCREG2
0x0A	OR	N/A	DESTREG ← SRCREG1 OR SRCREG2
0x0B	NOT	N/A	DESTREG ← NOT SRCREG1
0x0C	LSL	N/A	DESTREG ← LSL SRCREG1
0xOD	LSR	N/A	DESTREG ← LSR SRCREG1
0x0E	BRA	IM	PC ← Value
0x0F	BEQ	IM	IF Z=1 THEN PC ← Value
0x10	BNE	IM	IF Z=0 THEN PC ← Value
0x11	CALL	IM	M[SP] ← PC, SP ← SP – 1, PC ← Value
0x12	RET	N/A	SP ← SP + 1, PC ← M[SP]

Figure 2: Operation Codes

Furthermore, in order to perform that 19 different operations we required 24 different micro-instructions. Tables at the end of the file Table 1 and 2 shows the micro-operations and their objective; table 3,4,5 shows the Symbolic micro-program of our design.

In addition we designed our ROM to have 24 bit data in each address and 8 bit address values, we separated those 24 bit data as follows: first 4 bit for F1, second 4 bit for F2, third 4 bit for F3, fourth 4 bit for CD+BR, and the last 8 bit for address values. During our design we did not require the F3 operation field however if in the future the design requires additional micro-operations the field can be easily used since its completely empty. Furthermore designing the field bits are address values and CD+BR field in 4 bit or 8 bit gives us a clean ROM values since the every 4 bit is represented as hex values in ROM first bit in the ROM means F1 field in hexadecimal form, this form can be easily debugged and read since it is not needed to convert whole value to binary from in order to read specific field from the value. In addition, when designing CD and BR we used the lessons slide as a base however since the design of the cpu is different we omitted the unnecessary pieces and left them as blank so if in the future they are needed, they can be implemented.

3 RESULT

As a result of the project, we verified our implementation using the test code which is given in the project pdf file and with our own tests. There was not any contradiction between the expected and obtained results. Test code that we have tried over our implementation in the project pdf file is shown in the Figure 3.

```
        ORG 0x20            # Write the program starting from the address 0x20
        LD R0 IM 0x05       # R0 is used for iteration number
        LD R1 IM 0x00       # R1 is used to store total
        LD R2 IM 0xA0
        MOV AR R2           # AR is used to track data adrress: starts from 0xA0
LABEL:  LD R2 D             # R2 <- M[AR] (AR = 0xA0 to 0xA4)
        INC AR AR           # AR <- AR + 1 (Next Data)
        ADD R1 R1 R2        # R1 <- R1 + R2 (Total = Total + M[AR])
        DEC R0 R0           # R0 <- R0 – 1 (Decrement Iteration Counter)
        BNE IM LABEL        # Go back to LABEL if Z=0 (Itertaion Counter > 0)
        ST  R1 D            # M[AR] <- R1 (Store Total at 0xA5)
```

Figure 3: Test code from the project file

Our design is shown as in the Figure-4.

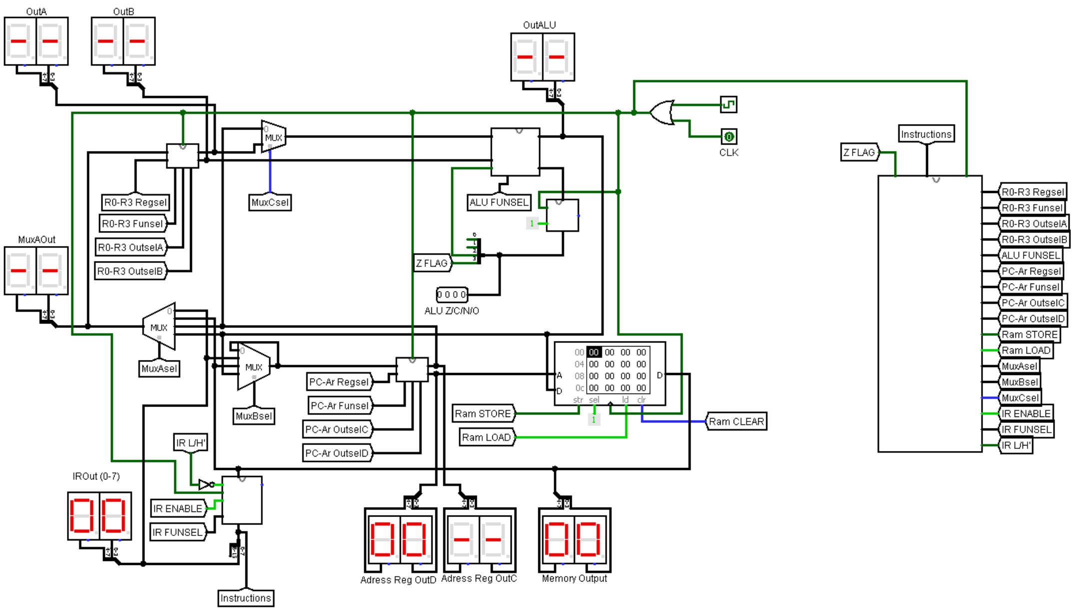

Figure 4: Final Implementation of Our Design

4 DISCUSSION

In this final project, we were expected to implement a software-control unit using our project 1&2 implementations and a few design ideas from project 3. We had micro-instructions and operations to deal with.Tackling new problems that rose with micro-operation based design was challenging at first. We have a few assumptions, for example register call for PC having 2 different codes was problematic for us. At first we tried using an OR gate to accommodate two codes representing PC but that proved to be wrong, OR gate would give an error when it had two X's as inputs. In addition when mapping our opcode we used the following format "1 opcode 00", as mentioned in the Design part last 2 bits represent how many address we will have when performing that operation which is 4, adding 1 to the first bit in the address allows us the keep the fetch micro-operation at the address 00000000 which means that when the project file is opened control address register will show the address 00000000 which means that the CPU will do the fetch operation next. Also if somehow CAR shows an unused address from ROM the hexa value of 000000 will be performed which means NO operation unconditional jump to 00000000 which is the fetch line meaning that the system will recover in case of an error as such

5 CONCLUSION

In conclusion this project helped define the idea of a software-control unit in our minds. Specifically how different it is to a hardwired counterpart. We had to use a completely different approach to our OPCODE design both in taking and executing the micro instructions. Instead of a time slot based on clock we used micro-operations from our ROM, which meant we were able to do similar tasks with way less wiring and gates, effectively cutting down total part cost.

6 APPENDIX

F1	Microoperation	Symbol	Description
0000	NONE	NONE	NONE
0001	Reg<-Reg	MOV	MOVE
0010	Rx<-IR(0-7)	LD	LOAD
0011	PC<-IR(0-7)	LPC	LOAD to PC
0100	AR<-IR(0-7)	LAR	LOAD to AR
0101	Reg<-M(AR)	RD	READ, Load from memory
0110	M(AR)<-Rx	STR	Store, write to memory
0111	M(SP)<-PC	CLL	CALL
1000	PC<-M(SP)	RTR	RETURN
1001	M(SP)<-Reg	PSH	PUSH
1010	Reg <-M(SP)	PLL	PULL
1011	AReg<-AReg+1	SIN	(Self increment) INCREMENT DESTREG
1100	AReg<- AReg-1	SDC	(Self decrement) DECREMENT DESTREG
1101	SP<-SP-1	DSP	DECREMENT SP
1110	SP<-SP+1	ISP	INCREMENT SP

Table 1: Microoperation table of F_1

F2	Microoperation	Symbol	Description
0000	NON	NON	NONE
0001	Reg<-Reg+Reg	ADD	ADD
0010	Reg<-Reg-Reg	SUB	SUB
0011	Reg<-Reg & Reg	AND	AND
0100	Reg<-Reg | Reg	OR	OR
0101	REG<- NOT REG	NOT	NOT
0110	REG <- LSL REG	LSL	LOGİCAL LEFT SHİFT
0111	REG<- LSR REG	LSR	LOGİCAL right shift
1000	IR(8-15)<-M(PC)	FUP	FETCH UP
1001	IR(7-0)<-M(PC)	FDW	FETCH DOWN
1010	PC<-PC+1	IPC	INCREMENT PC

Table 2: Microoperation table of F_2

ROM address	Microop	F1	F2	F3	CD-BR	Address field
00000000	FETCH UP	0000	1000	0000	0000	00000001
00000001	INC PC	0000	1010	0000	0000	00000010
00000010	FETCHDOWN	0000	1001	0000	0000	00000011
00000011	INC PC	0000	1010	0000	0011	XXXXXXXX
10000000	NON	0000	0000	0000	0100	10000010
10000001	LOAD	0010	0000	0000	0000	00000000
10000010	READ	0101	0000	0000	0000	00000000
10000011	UNUSED					
10000100	NON	0000	0000	0000	0100	10000110
10000101	NON	0000	0000	0000	0000	00000000
10000110	STORE	0110	0000	0000	0000	00000000
10000111	UNUSED					
10001000	MOVE	0001	0000	0000	0000	00000000
10001001	UNUSED					
10001010	UNUSED					
10001011	UNUSED					
10001100	DEC SP	1101	0000	0000	0000	10001101
10001101	PUSH	1001	0000	0000	0000	00000000
10001110	UNUSED					
10001111	UNUSED					
10010000	PULL	1010	0000	0000	0000	10010001
10010001	INC SP	1110	0000	0000	0000	00000000
10010010	UNUSED					
10010011	UNUSED					
10010100	ADD	0000	0001	0000	0000	00000000
10010101	UNUSED					
10010110	UNUSED					
10010111	UNUSED					
10011000	SUB	0000	0010	0000	0000	00000000
10011001	UNUSED					
10011010	UNUSED					
10011011	UNUSED					

Table 3: Symbolic Microprogram-1

ROM address	Microop	F1	F2	F3	CD-BR	Address field
10011100	MOV	0001	0000	0000	0000	10011101
10011101	SELF DEC	1100	0000	0000	0000	00000000
10011110	UNUSED					
10011111	UNUSED					
10100000	MOV	0001	0000	0000	0000	10100001
10100001	SIN	1011	0000	0000	0000	00000000
10100010	UNUSED					
10100011	UNUSED					
10100100	AND	0000	0011	0000	0000	00000000
10100101	UNUSED					
10100110	UNUSED					
10100111	UNUSED					
10101000	OR	0000	0100	0000	0000	00000000
10101001	UNUSED					
10101010	UNUSED					
10101011	UNUSED					
10101100	NOT	0000	0101	0000	0000	00000000
10101101	UNUSED					
10101110	UNUSED					
10101111	UNUSED					
10110000	LSL	0000	0110	0000	0000	00000000
10110001	UNUSED					
10110010	UNUSED					
10110011	UNUSED					
10110100	LSR	0000	0111	0000	0000	00000000
10110101	UNUSED					
10110110	UNUSED					
10110111	UNUSED					
10111000	NON	0000	0000	0000	0100	00000000
10111001	LPC	0011	0000	0000	0000	00000000
10111010	UNUSED					
10111011	UNUSED					

Table 4: Symbolic Microprogram-2

ROM address	Microop	F1	F2	F3	CD-BR	Address field
10111100	NON	0000	0000	0000	0100	00000000
10111101	NON	0000	0000	0000	1100	10111111
10111110	NON	0000	0000	0000	0000	00000000
10111111	LPC	0011	0000	0000	0000	00000000
11000000	NON	0000	0000	0000	0100	00000000
11000001	NON	0000	0000	0000	1100	00000000
11000010	LPC	0011	0000	0000	0000	00000000
11000011	UNUSED					
11000100	NON	0000	0000	0000	0100	00000000
11000101	DEC SP	1101	0000	0000	0000	11000110
11000110	CALL	0111	0000	0000	0000	11000111
11000111	LPC	0011	0000	0000	0000	00000000
11001000	RETURN	1000	0000	0000	0000	11001001
11001001	INC SP	1110	0000	0000	0000	00000000
11001011	UNUSED					
11001100	UNUSED					

Table 5: Symbolic Microprogram-3