ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 222E

COMPUTER ORGANIZATION PROJECT-3 REPORT

GROUP MEMBERS:

150170022	:	FURKAN HAYTA
150170038	:	CİHAN ALPER ERTEM
150170061	:	MUHAMMED FURKAN DEMİR
150170092	:	BARIŞ İNCESU

SPRING 2020

Contents

1 INTRODUCTION

The final version of the project, which started with project 1, is expected to be completed in project 3. If a brief summary of the project is required, Register File, Address Register File and Instructor Register were installed in the first stage shown in figure 1. Subsequently, in Project 2, Arithmetic Logic Unit (ALU) has been designed as the computational element of the system shown in figure 1. In this project, the last main unit of the system, the Control Unit, will be created and Memory - ALU - Registers will be connected. The 18 operations expected to be performed by the Central Unit are shown in the table in figure 2.

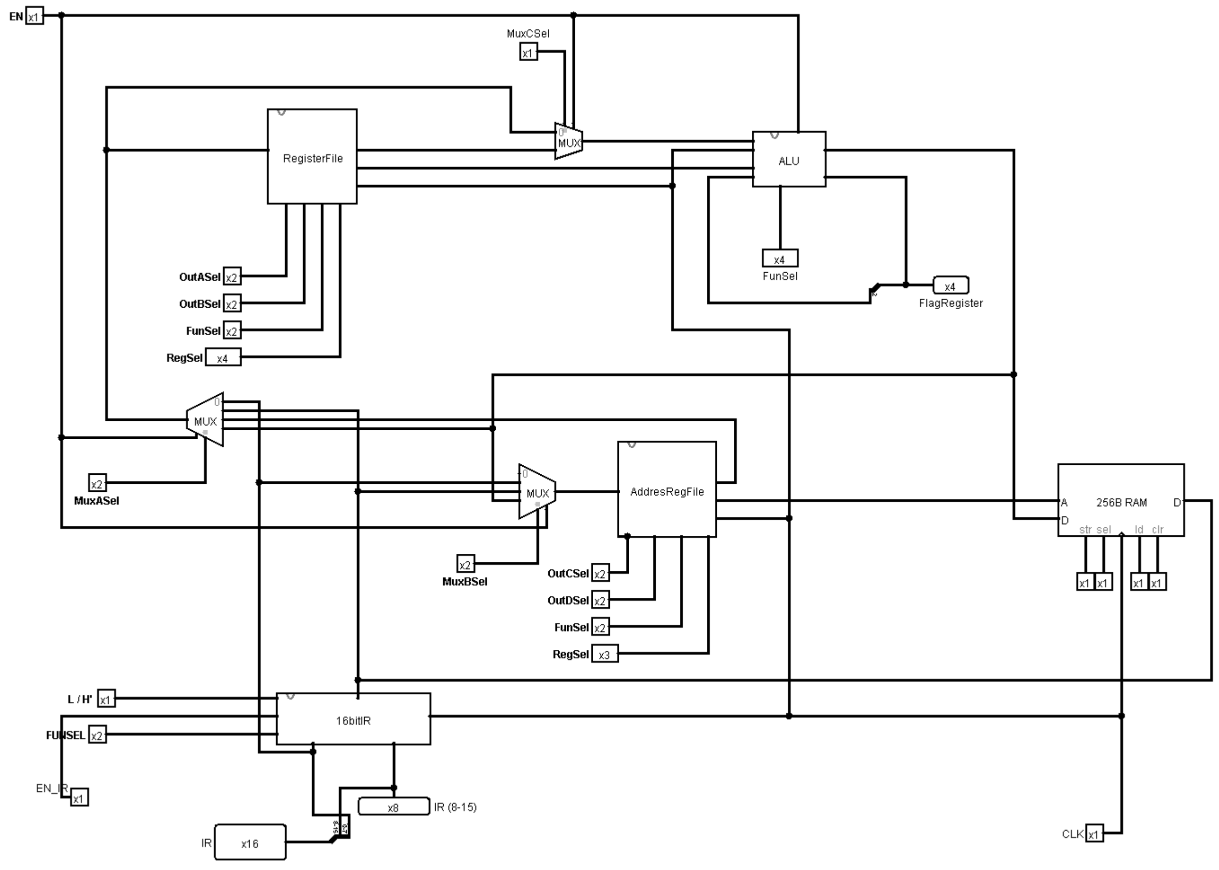
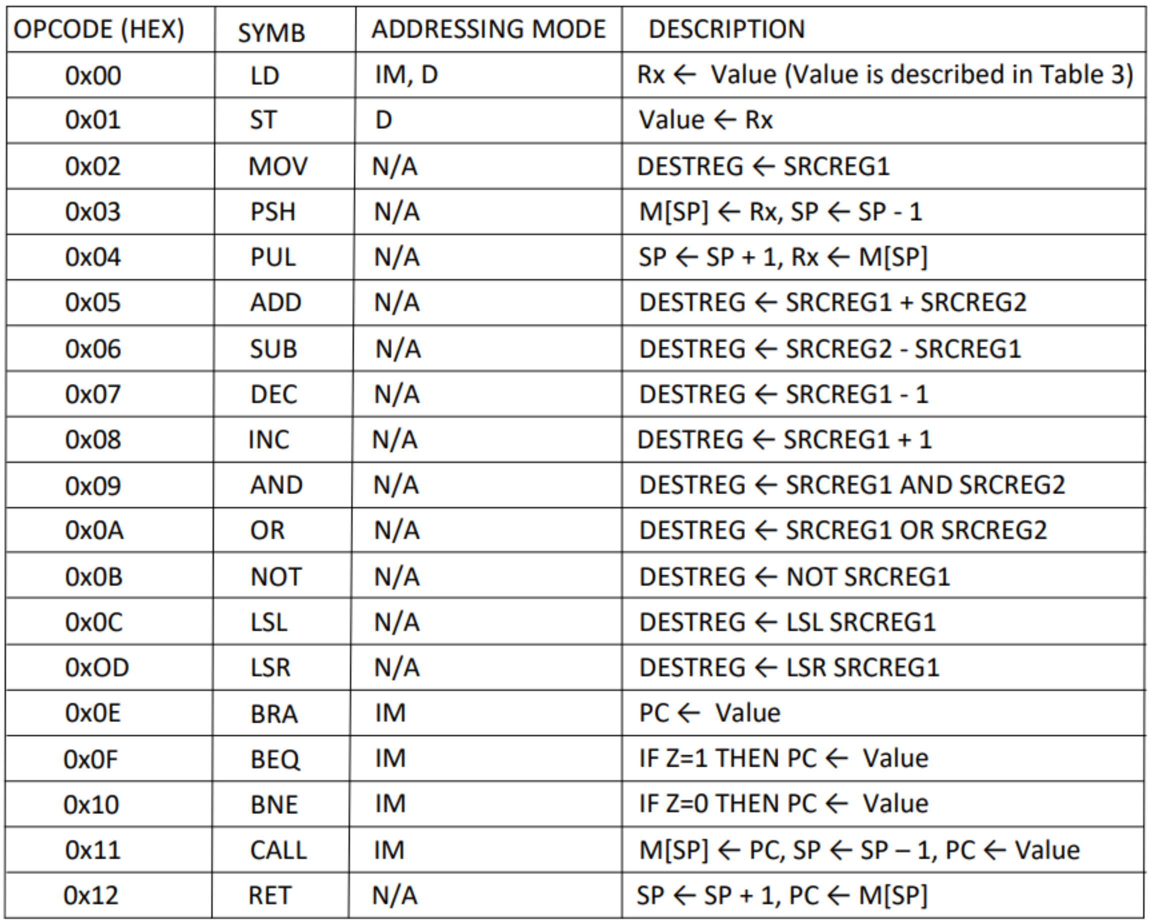


Figure 1: System Organization



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: Central Unit Operations

2 METHODOLOGY

In this project, operations are performed by assigning the necessary values to the systems created in project 1 and project 2 in general. As can be seen in Figure 2, 16 bit numbers given over RAM are made piece by piece and necessary operations are carried out in the circuit. The section called OPCODE (the first 5 bits of the input) provides the selection of these processes. OPCODE is inserted into a decoder to enable process selection in the circuit. Its outputs are also named and tunneled, respectively. For example, if OPCODE = 00000 it will take D0 = 1 input (fig3).

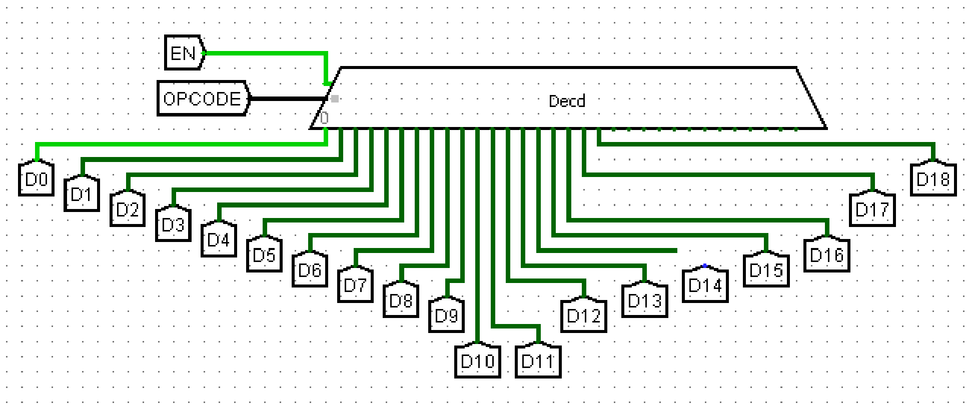


Figure 3: D Chooser

However, the D system can only select the required process, and when the process is selected, different inputs are activated for each clock. To achieve this, we built a T system. In this system, the T value changes in every full turn of the clock, and this t value acts as a clock and decides when and in what order the operations to be performed (with the help of D) (fig4).

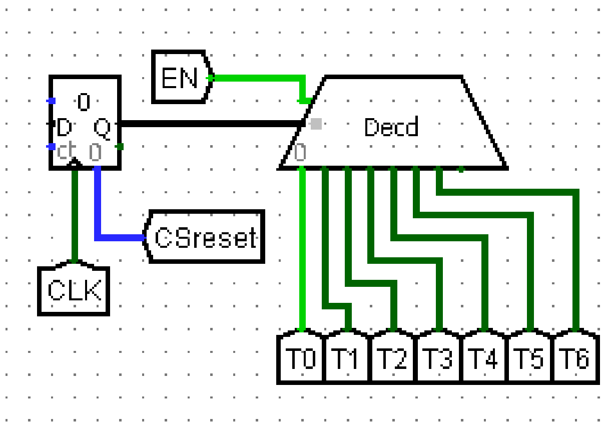


Figure 4: T Chooser

The following are the specific OPCODE choices and the transactions made as a result of the elections. Certain processes are the same for all OPCODE values, some processes are the same for some OPCODE values. In order to prevent transaction crowd, these situations were bridged during the planning phase and reported as such. Logisim version 2.7.1 was used in the construction, circuit drawings and diagramming of this assignment.

2.1 T0, T1 (ALL OPCODES) and T2 (with address)

There are always some things to do when any OPCODE value is added to the circuit. These are taking the value from Ram, writing it to IR, assigning it to the necessary points for the expected operations that are broken down and disabled, and assigning an address to the RAM for the next clock. These situations are provided in T0 and T1. However, exceptionally, some inputs (Adressing mode: IM) are done with another clock.

2.1.1 T0

At T0, the following processes occur for all OPCODE processes:

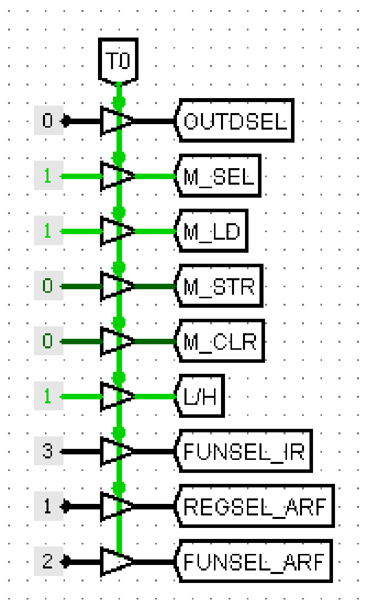


Figure 5: T0

The first 8 bits of the value in the RAM are taken thanks to the above processes, and they are loaded into the High part of the IR. Increasing the value of the PC in the Adress Register File by 1, it prepares to read the next value in RAM at the time of T1.

2.1.2 T1

At T1, the following processes occur for all OPCODE processes:

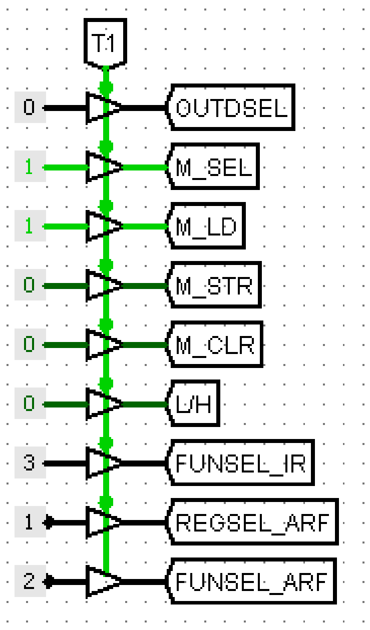


Figure 6: T1

The least 8 bits of the value in the RAM are taken thanks to the above processes, and they are loaded into the low part of the IR. Increasing the value of the PC in the Adress Register File by 1, it prepares to read the next value in RAM at the time of T2. As a result of making T1, it is written to 16 bit input IR and the required operations are customized with OPCODE in clocks that progress with the help of three state buffers.

2.1.3 T2 (with address)

At T1, the following processes occur for only 2 OPCODE processes (those with addressing mode D):

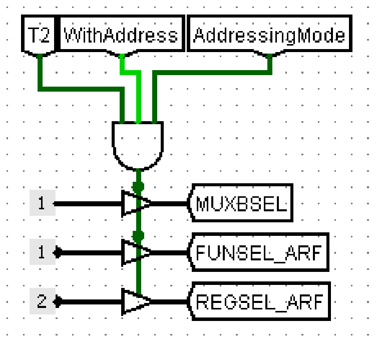


Figure 7: T2 with adress

Here, the Low part of the value to IR is loaded into the AR in the Adress Register File thanks to MuxB, and the value at that address can be extracted from RAM and used in the next operation.

2.2 0X00 LD OPERATION

For load operation , instruction is decoded according to instructions with an address reference .(Aşağı ya da yukarıdaki tabloda).Instructions with an address reference start to execute at T3 clock signal.At T3, Since the value has been defined, Values is loaded to Rx. Rx is determined by Regsel.($00- > R_1, 01- > R_2, 10- > R_3, 11- > R_4$). .Value comes from output of Memory or Instruction Register Low Bit depends on Addressing Mode. MuxA select one of these and sends to Register File. To load register in register file , FUNSEL of Register File must be 01 and Regsel (4-bit) of Register File is set depending on Rx. After these operation , value is loaded succesfully to Rx.

2.3 0X01 ST OPERATION

ST operation aims to store value of Rx at memory (M[AR]) .This is also operated by instructions with an address reference.Since the operation is storing at memory , Addressing Mode is only Direct. Data at Register File have to be transferred to D(data) input of Memory.AR is assigned at T2 like all direct address reference instructions.

T3 : Firstly, selected register in Register File have to transmit to ALU.We choose OUTBSEL to use for transmit. OUTBSEL is set as REGSEL that is 10-9 th bit of IR. Since data should not change , FUNSEL of ALU is set as 0001 ($B- > B$).After these processes, the data has reached the memory and now it is necessary to reach the address.AR is sent to memory through OUTDSEL. Therefore , OUTDSEL should be 01.If the str and sel entries of the memory is activated (Str=1 , Sel=1 , Ld=0 ,Clr =0) , this process is completed.

2.4 0X03 (PSH) OPERATION

In this process, T2 clock is taken to the ALU via OUTB by taking the value from the desired register in the register file, and it is loaded into the data arm of the RAM in the output without changing the value in the ALU. Lastly, the address information from the SP register in the address register file is sent to RAM with OUTD, and in RAM, the writing process is completed by sending one to the store input. At the time of T3 clock, 1 reduction is completed by selecting the SP register, which is the last operation, from REGSEL and sending the decrement position on FUNSEL.

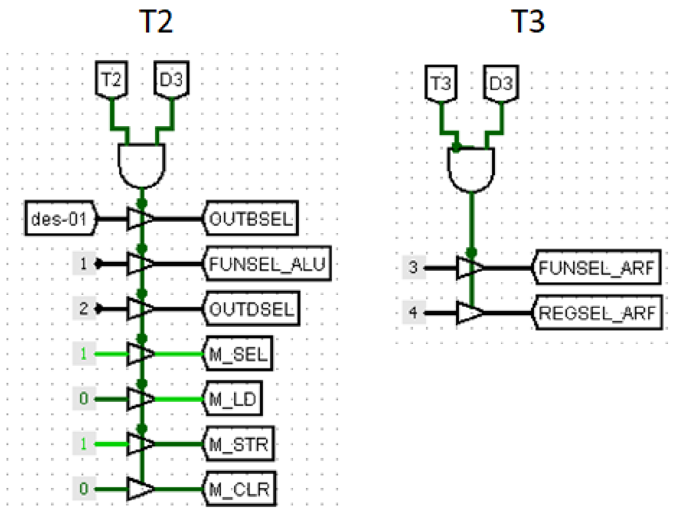


Figure 8: Push operation

2.5 0X04 (PULL) OPERATION

In this process, first of all T2 clock increment process was sent to SP register in Adress register over FUNSEL. Then, a T3 clock was sent to the selection and load inputs via RAM, and the value at the address SP + 1 is out of RAM. This value is completed on MUXA by clicking REGSEL with DESTREG and uploading function.

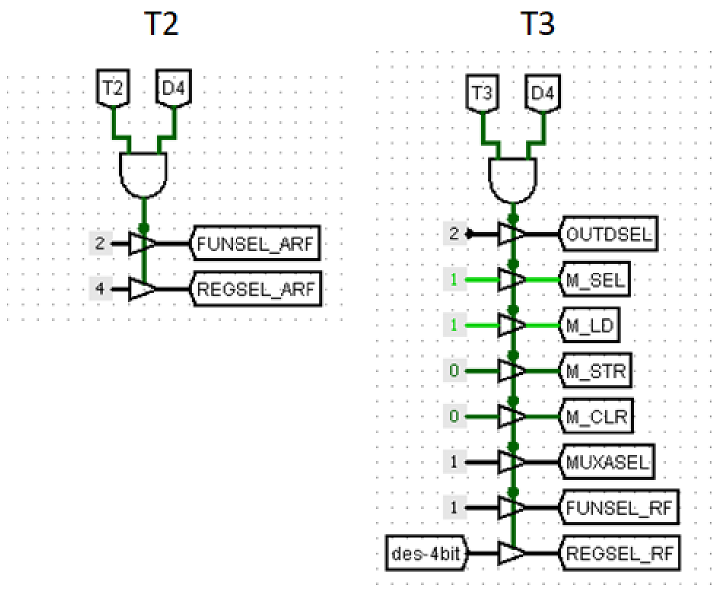


Figure 9: Pull operation

2.6 Transfer Operations From SRCREG1 to DESTREG

Since the general structure of these operations is almost the same, they are gathered under " Transfer Operations From SRCREG1 to DESTREG ".Structure and path will be represented in DESTREG ← SRCREG1. Differences or additional operations will be represented in its own part.

2.6.1 DESTREG ← SRCREG1 0X02 MOV

This operation aims to transfer data from source register to ,destination register.Instruction register without adress reference is decoded. SRCREG1 and DESTREG is defined.There are four different path from SRCREG1 to DESTREG.

- From Register File to Register File :
  Data is transferred from ALU by using OUTASEL. A → A function of ALU is operated.Data passes from MUXASEL and it reaches to Register File.Data is loaded to destination register that is pointed by DESTREG.

  OUTASEL = SRCREG1 [1:0]
  MUXCSEL = 1
  FUNSEL of ALU = 0000
  MUXASEL = 11
  FUNSEL of RF = 01
  REGSEL of RF = register that pointed by DESTREG

- From Register File to Address Register File

  Data is transferred from same path until reach to MUXASEL.Here , data passes from MUXBSEL instead of MUXASEL.Data that reaches to Address Register File is loaded to destination register that is pointed by DESTREG.

  OUTASEL = SRCREG1 [1:0]
  MUXCSEL = 1
  FUNSEL of ALU = 0000
  MUXBSEL = 11
  FUNSEL of ARF = 01
  REGSEL of ARF = register that pointed by DESTREG

- From Address Register File to Register File
  Now Source Register is from Address Register File.Transfer starts from OUTC. It reaches MUXA .Data passes through MUXA and it is loaded to register in Register File.

  OUTCSEL = SRCREG1[1:0] - 1
  MUXASEL = 10
  FUNSEL of RF = 01
  REGSEL of RF = register that pointed by DESTREG

- From Adress Register File to Address Register File
  Data that reaches to Register File is not loaded to RF.It goes to ALU and comes to MUXB without change.And data is loaded to the register in Address Register File.

  OUTCSEL = SRCREG1[1:0] - 1
  MUXASEL = 10
  MUXCSEL = 0
  FUNSEL of ALU= 0000
  MUXBSEL = 11
  FUNSEL of ARF= 01
  REGSEL of ARF = register that pointed by DESTREG

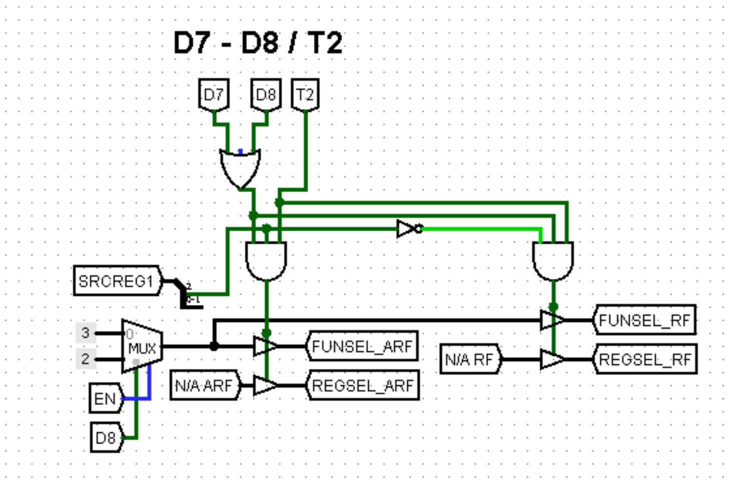
2.6.2 DESTREG ← SRCREG1 +1 / -1

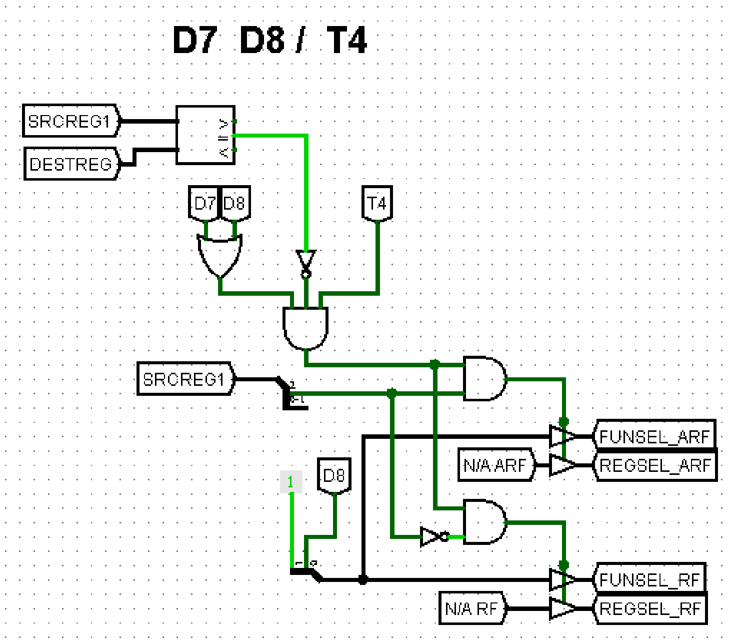
Except for a case, the same path as 0X02 is used.So data has to pass from ALU.From ARF to RF does not support this in 0X02 MOV operation.

OPCODE 0x07 :It aims to decrement data at ALU and transfer to destination Register.
OPCODE 0x08 :It aims to increment data at ALU and transfer to destination Register.

In order to perform operations mentioned above, it is necessary to assign 1 to B input of ALU.The only way to do this is to assign 1 to one of the registers in Register File. This can cause a significant value to be lost.In order to get Z flag correctly and not to change registers randomly, we thought of a solution like this : Firstly, source register is incremented one in its file.And data is transferred to ALU and A− >A function is operated. Data is loaded destination register . If destination and source register are different, the increased source register is reduced vice versa.

AT T2 : SRCREG1 is decremented for 0x07 operation and is incremented for 0x08 operation.
AT T3 : SRCREG1 is transferred to DESTREG .(0x02 MOV is used with a additional circuit for from ARF to RF .)
AT T4 : If destination and source register are different, SRCREG1 is incremented for 0x07 operation and is decremented for 0x08 operation.





2.6.3 DESTREG ← NOT SRCREG1 | LSL SRCREG1 | LSR SRCREG1

0X0B-NOT , 0X0C-LSR and 0X0D-LSR operations have nearly similar process.Only difference betweeen them is function of ALU.Data comes from SRCREG1 to ALU.Funsel is set according to desired operation and output of ALU transferred to DESTREG.These operations is done in same circuit with above.Since these logic and shift operations is done in ALU , additional circuit for 0X07 and 0X08 is used .

FUNSEL of ALU is 0010 for 0X0B-NOT
FUNSEL of ALU is 1010 for 0X0C-LSL
FUNSEL of ALU is 0011 for 0X0D-LSR

2.7 0X05 (ADD), 0X06 (SUB), 0X09 (AND), 0X0A (OR)

The reason why these four processes are common in this section is that they choose the function belonging to their own transactions in ALU in the same circuit and give the result correctly.

In this section, the circuit is completed with a total of four clock cycle. Two common clocks (T0, T1) are explained first part of methodology, for all operations. T2 and T3 clock operations will be explained below. However, there are transactions with 2 numbers in this section and the number of transactions can increase according to the location of the numbers. While T2 is sufficient for some inputs, T3 can be used for some inputs.

2.7.1 T2:

At T2, the location of these values is determined by looking at the MSB of the 2 inputs (SRC1 and SRC2) selected according to the value recorded in the IR. As a result, 3 different situations arise.

- SRC1 = Register file, SRC2 = Register file
- SRC1 = Register file, SRC2 = Address register file
- SRC1 = Adress register file, SRC2 = Register file

Adress register file to Adress register file status is not added because it is not requested in the project. In addition, since the location of SRC1 and SRC2 is important in the SUB case (OPCODE 0X06), the second and third cases are examined separately.

These situations are also divided into two:

- DESTREG = Register file
- DESTREG = Adress register file

As a result, 6 different paths are created in the circuit with the 16-bit command received from the IR. While all operations can be combined in 4 ways at T2, T3 is required in 2 ways. For example, in the circuit shown below, only T2 is sufficient for all operations. (The rest of the circuit is not included in the Figure to avoid confusion.)

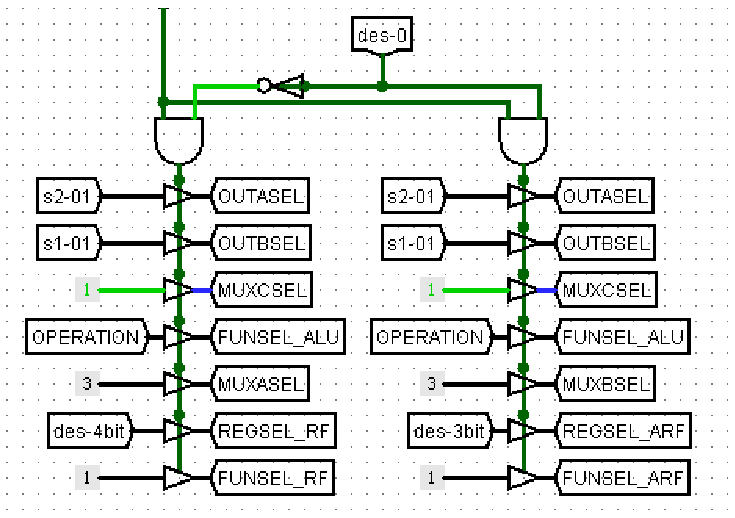


Figure 10: SRC1 = RF, SRC2 = RF

From the values in the register file at the beginning of the process, the output selection was made first to send SRC2 to OUTA to SRC1 is OUTB. Then, 1 operation was selected from MUXC and the values were sent to ALU. MUXA has been selected since it will bring the output from ALU back to the register file. Finally, the register selection process and the loading function in the register file have been applied to the circuit and a single clock operation has been completed.

2.7.2 T3:

In some cases, T3 was needed because T2 was not sufficient to perform procedures.The structure below is one of the points where T3 is needed.

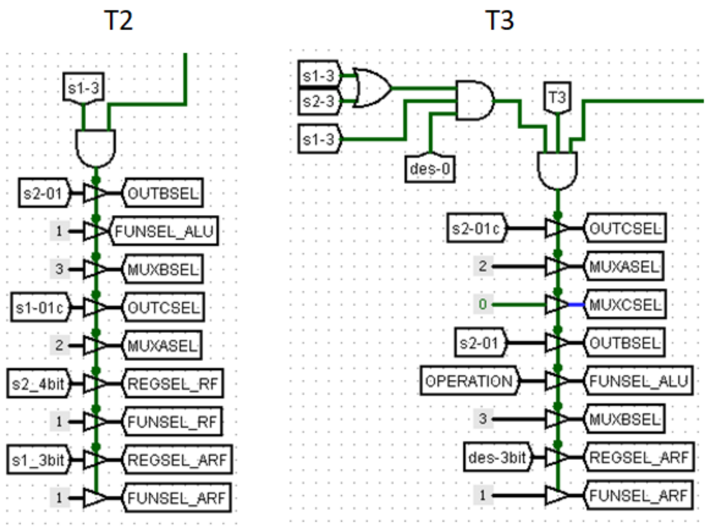


Figure 11: SRC1 = ARF, SRC2 = RF

Since SRC1 is in register file while SRC1 is in register file, we will start this process by writing SRC1 to SCR1 recorder in address register file instead of SRC2 in register file, since we will use this circuit in SUB operation. With this method, we have found a solution for A-B operation in ALU in SUB operation even though there is no loss of value. T2 clock process is finished here. Then we wrote SRC2 to register file via MUXA and sent SRC1 to OUTB, SRC2 to OUTA and ALU. We have completed the T2 clock process by selecting the loading operation on the output of MUXB by the register selected by DESTREG.

The circuit we designed is performing 4 operations (ADD-SUB-AND-OR) in the same place. The process separation is made with the help of Operation selector. The values assigned to the operation input and the operations performed are as follows:

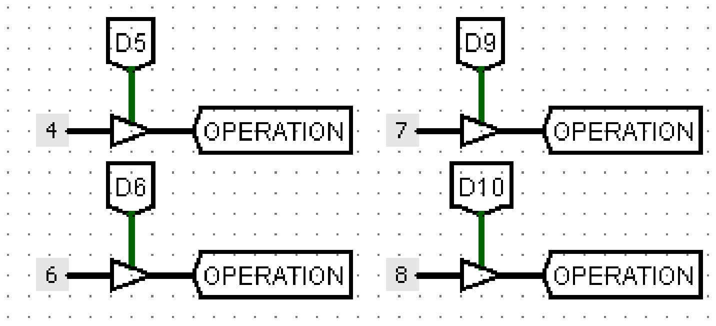


Figure 12: Operation selector

- D5 = ADD, operation = 0100
- D6 = SUB, operation = 0110
- D9 = AND, operation = 0111
- D10 = OR, operation = 1010

The operations required for all inputs are applied as in the examples. T2 and T3 processes are completed immediately. In the next clock, only Counter selection is reset and a single circuit is installed for this operation.

2.8 0x0E BRA, 0x0F BEQ, 0x10 BNE

Since these D14, D15, D16 coded operations are very similar, they are handled together and solved in a single circuit. Step-by-step operations are specified according to the Clock steps.

Since the AdressingMode is specified as IM, it should work in cases where AdressingMode enters 0. Accordingly, it was added to the circuit by not taking the AdressingMode in T2 stage.

The other components of this circuit are the Value value that comes depending on the operation codes. Operations of MUXBSEL 01 (Memory Input), RegSel 001 (ONLY PC ENABLED) and FunARSel 01 (Load) were applied to load this value to the PC.

MSel and MLoad values are opened to extract appropriate values from memory.

Z values were withdrawn from FLAG and integrated into the circuit with appropriate conditions.

2.9 0x11 CALL

The left-to-right path was followed for the order of the operations expected in D17. Accordingly, the Clock steps have been decided and implemented.AddressingMode value must be 0, so all steps are added.

T2: PC value must be assigned to M [SP] value. For this, the address input to be entered into the Memory must show M [SP] and the data must show the value of the PC. The circuit takes the first clock step accordingly. It takes OUTDSEL (10) and sends the value of SP to Memory as Adress.

It outputs the address of the PC from the Address File Register with the value OUTCSEL (00). In order for this value to enter the D input of the Memory, it first comes to ALU as MUXCSEL (0) after MUXASEL (10), and in ALU it passes directly with ALUFunSel (0000) without processing.

T3: Memory STR and Selection are turned on because the values are ready to be written. Thus, M [SP] ¡- PC operation was completed. In order to be SP ¡- SP - 1, RegSel 11 from Address Reg File (ONLY SP ENABLED) and Decrement operation FunSel (100) are entered.

T4: The last remaining PC ¡- Value was set to MUXBSEL (01) to get the value from the IR for operation. Later, RegSel 001 (ONLY PC ENABLED) and FunSel Load (01) were assigned Value to PC.

2.10 0x12 RET

T2: D18 started the operation with SP ¡- SP + 1 operation. For this stage, FunSel 10 (Incremeny) and RegSel 100 (ONLY SP ENABLED) were made from Address Reg File.

T3: In this clock step, the generated SP value is sent to Memory with OUTDSEL 10. The address is reached with the Memory Selection section and the value is drawn with Memory Load. This value was passed to MUXBSEL10 and brought to the Address Register File. It remains only to assign this value to the PC. This can be done with RegSel 01 (ONLY PC ENABLED) and FunSel 01 (Load).

2.11 SC RESET

The circuit below ensures that the T counter is reset when the last T value is reached for all OPCODE states. In this way, the new processes directory is passed without an empty clock and the system continues to operate in a healthy way. These mini processes are gathered in a single circuit to prevent confusion in the system.

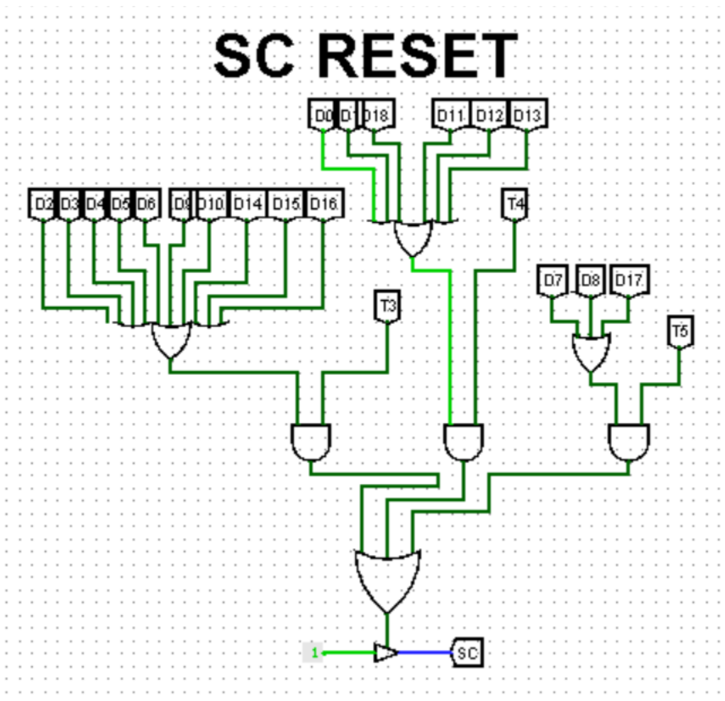


Figure 13: T Reset System

3 RESULTS

As a result of this circuit that we have designed the hardwire-control unit, the instructions coming with or without address from RAM are written in the IR register section. Here, the structure of deciding which operations to perform by breaking into pieces was observed. Some transactions are shown below with their results.

EXAMPLE :

In this example, we print the value 7 on the R1 register and add it from the PC register with the value 4, and print the result to the SP address in RAM.

Printing to R1: 0000001000000111
R1 and PC picking process: 0010111100110000
Installing the RAM onto the SP address:00011001

As in this example, expected results from the circuit were observed in other examples. No problems were encountered.

4 DISCUSSION

In this project, we designed the internal structure of a hardwire-control unit to the finest detail. With this structure, we learned a lot about getting value from RAM and processing the received values and at the same time, we developed ourselves in this regard. This project gave us an idea of how the orders placed in the internal structure of a computer are implemented, the codes entered, and how the electrical hardware rotating in the background of these codes works. We also learned that we can design more complex circuits using circuit elements such as three-state buffer and tunnel needed in this project. Thanks to this project at the same time, we realized how important team work is in big projects. Each team member helped each other in the installation and testing of the designed circuits, so that the project could be completed completely. Thanks to the last project, we realized that the circuits and structures designed in project 1 and project 2 are parts that perform important operations in a computer.

5 CONCLUSION

Despite many difficulties at the end of Project 3, many teachings were gained. The project, which started with and and or gates, resulted in a mini computer. If we touch on the difficulties and achievements at the end of this project, it is clear that the biggest challenge and achievement is teamwork. The magnitude of the importance of team work has increased in each project compared to other projects. The confusion within the team and the incompatibilities that have occurred in some times have been successfully resolved through in-team meetings. The progress of the structures mentioned in the lectures from the simplest to the most complex with this project series, step by step, helped to learn what each unit does. In the last project, the project was completed on time even though some defects in the SUB process and some uncertainties in debugging the Clock step model.

6 References

[1] https://tr.overleaf.com/project

[2] https://www.studytonight.com/computer-architecture/addressingmodes-instructioncycle

[3] BLG 222E COMPUTER ORGANIZATION Week 5-7 Slides