# Department of

Computer Engineering

## BLG 222E

## Computer Organization

## Report of Project-1

### Experiment Date: 15.03.2018

Member's ID : Member's Name :

150150007 Hatice Hüma Kalaycı

150150801 Alican Kurt

150150067 Refik Özgün Yeşildağ

150140061 Hakan Saraç

### **1. Introduction**

The first project of Computer Organization course wants from us to design 8-bit and 16-bit registers, a general purpose register and an 16-bit IR adress register. We designed the our systems as follows.

### **2. Equipments and Integrated Circuits**

When preparing the circuits, we used these equipments in Logisim.

* 1-bit adders
* 2-select 8-data bits Multiplexer
* D flip-flops
* 8-bit registers
* 16-bit register

### **3. Experiment**

### **3.1. Part-1**

Control inputs and functions:

|  |  |  |  |
| --- | --- | --- | --- |
| Enable | FunSel | Q⁺(Next State) | Function |
| 0 | **Ø** | **Q** | **Retain Value** |
| 1 | **00** | **0** | **Clear** |
| 1 | **01** | **Q+1** | **Increment** |
| 1 | **10** | **Q-1** | **Decrement** |
| 1 | **11** | **I(Input)** | **Load** |

*Table 3.1.a Funsel control inputs and Functions*

### **3.1.1. 8-bit Register**

In this part we designed a 8-bit register which performs the clear, increment, decrement and load functions. To increment we added 1, and to decrement added (-1) 1111(2's complement of 1) to the our last state. Our FunSel inputs select the multiplexer output. We use enable input instead of clock for the D flip-flops.

### **C:\Users\Hakan\AppData\Local\Microsoft\Windows\INetCache\Content.Word\8bit.png**

### 

### *Figure 3.1.1.a 8-bit Register*

### **3.1.2. 16-bit Register**

### **C:\Users\Hakan\AppData\Local\Microsoft\Windows\INetCache\Content.Word\16bit.png**

### *Figure 3.1.2.a 16-bit Register*

In this step we use 16-bit instead of 8-bit. Its working principle is the same as 8-bit register.

### **3.2. Part-2**

### **3.2.1. Part-2.a System of four 8-bit general purpose registers**

Control inputs and functions:

|  |  |  |  |
| --- | --- | --- | --- |
| Enable | FunSel | Q⁺(Next State) | Function |
| 0 | **Ø** | **Q** | **Retain Value** |
| 1 | **00** | **0** | **Clear** |
| 1 | **01** | **Q+1** | **Increment** |
| 1 | **10** | **Q-1** | **Decrement** |
| 1 | **11** | **I(Input)** | **Load** |

*Table 3.2.1.a Funsel control inputs and Functions*

|  |  |
| --- | --- |
| OutAsel | Output(A) |
| 00 | **R0** |
| 01 | **R1** |
| 10 | **R2** |
| 11 | **R3** |

|  |  |
| --- | --- |
| OutBsel | Output(B) |
| 00 | **R0** |
| 01 | **R1** |
| 10 | **R2** |
| 11 | **R3** |

*Table 3.2.1.b Outsel control*

|  |  |
| --- | --- |
| Regsel | Enabled Registers |
| 0000 | **-** |
| 0001 | **R3** |
| 0010 | **R2** |
| 0011 | **R2,R3** |
| 0100 | **R1** |
| 0101 | **R1,R3** |
| 0110 | **R1,R2** |
| 0111 | **R1,R2,R3** |
| 1000 | **R0** |
| 1001 | **R0,R3** |
| 1010 | **R0,R2** |
| 1011 | **R0,R2,R3** |
| 1100 | **R0,R1** |
| 1101 | **R0,R1,R3** |
| 1110 | **R0,R1,R2** |
| 1111 | **R0,R1,R2,R3** |

*Table 3.2.1.c Outsel inputs and Outputs*

### **C:\Users\Hakan\AppData\Local\Microsoft\Windows\INetCache\Content.Word\part2a.png**

### 

### *Figure 3.2.1.a 8-bit general purpose registers*

We used 4 8-bit registers which is designed in part-1. Our inputs are clear, load, enable, FunSelect and RegSelect. Register Select decide which registers work (Table 3.2.1.c). FunSel inputs select the function (Table 3.2.1.a).

### **3.2.2. Part-2.b 8-bit adress registers**

Control inputs and functions:

|  |  |
| --- | --- |
| OutCsel | Output(C) |
| 00 | **PC** |
| 01 | **PC** |
| 10 | **AR** |
| 11 | **SP** |

*Table 3.2.2.a OutCsel control*

This circuit performs the functions as Part2-a. But both 00 and 01 OutCSel inputs give the same result (PC). If OutCSel inputs are 10, we get AR's result and if they are 11, we get SP's result. Our output shows us the selected registers last state.

### **C:\Users\Hakan\AppData\Local\Microsoft\Windows\INetCache\Content.Word\part2b.png**

### *Figure 3.2.2.a 8-bit adress registers:PC, AR, SP*

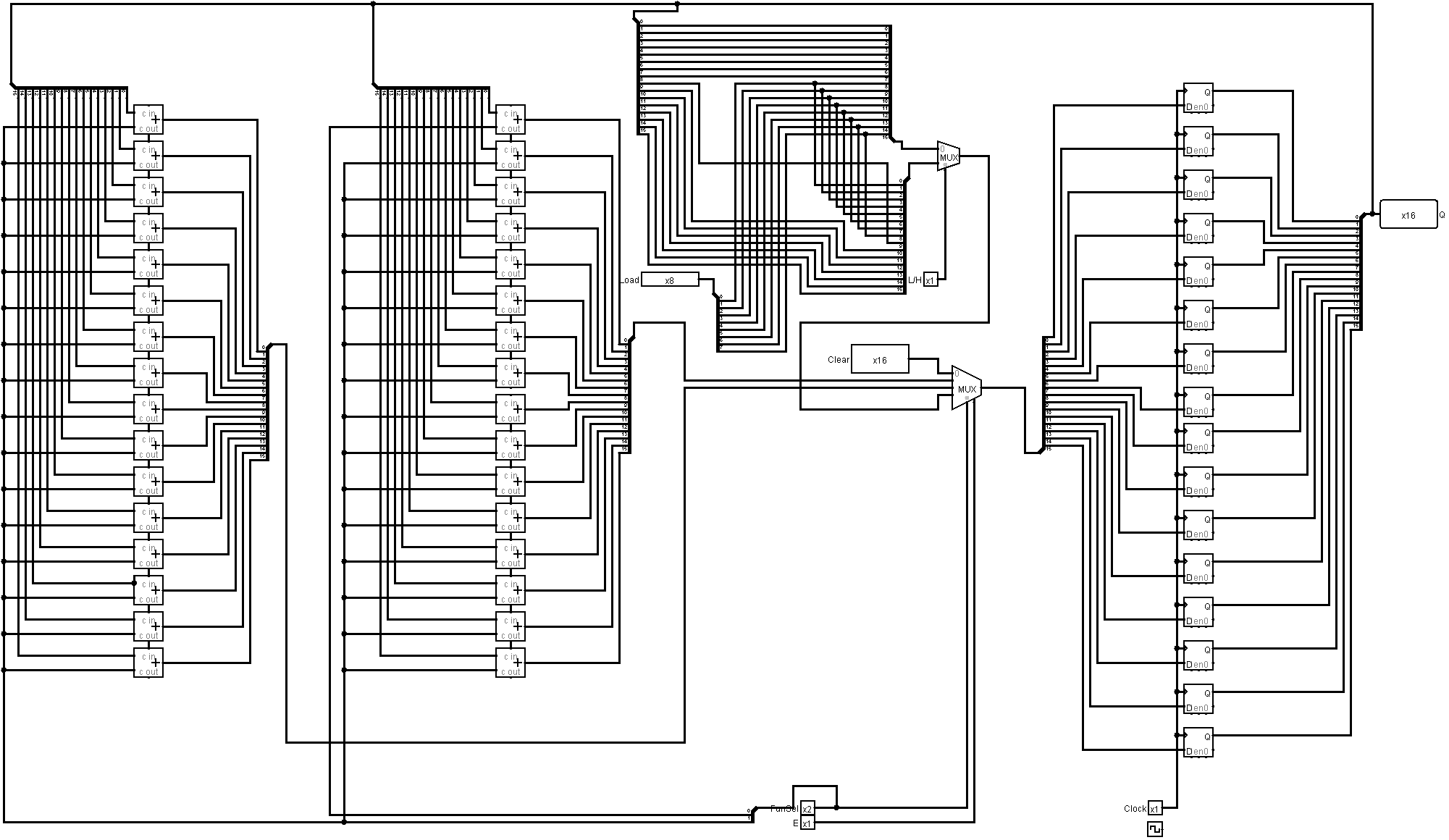
### **3.2.3. Part-2.c 16-bit IR register**

Control inputs and functions:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| L | Enable | FunSel | IR⁺ | Function |
| Ø | **0** | **Ø Ø** | **IR** | **Retain Value** |
| 1 | **1** | **11** | **IR(0-7)<-I** | **Load 0-7 bits** |
| 0 | **1** | **11** | **IR(8-15)<-I** | **Load 8-15 bits** |
| Ø | **1** | **00** | **0** | **Clear** |
| Ø | **1** | **01** | **IR+1** | **Increment** |
| Ø | **1** | **10** | **IR-1** | **Decrement** |

*Table 3.2.3.a Characteristics of IR register*

When we were designing the 16\_bit IR register, we used the 16-bit register which is in the part-1. But our 8-bit load value replace registers 0-7 or 8-15 bit parts. Part replaced is selected by L input. If L is 0 our value replace 8-15 bit, otherwise 0-7 bit. When we decide the part, we use a multiplexer 1 select(L) and 16 data bits.



### *Figure 3.2.3.a 16-bit IR register*

### **4. Conclusion**

In this project we learned using Logisim Logic Simulator Software and designing and using registers by this program. All of team members ready for next subjects and projects.