**Note: RAM’den read yapmak için clk ya gerek yok, address e strore etmek için var**

**Opcode0:** Rx ← Value

Note: Addressing mode IM, D

**IR8 = 0**

**Update:**

Opcode0T4: **Rx 🡨 IROut(0,7)**

MuxASel = 00

FunSel(2A) = 01

RegSel(2A) = REGSEL-2A-0

**IR8 = 1**

Opcode0T4: **Rx 🡨 M[AR]**

OutDSel = 01;

Read = 1;

Load = 0;

MuxASel = 01;

OutASel = IR(9,10);

FunSel(2A) = 01;

RegSel2A = REGSEL-2A-0;

**Opcode1:** Value ← Rx

Note: Addressing mode D

Opcode1T4: **M[AR] 🡨 Rx**

RegSel(2A) = regsel-2a-0;

OutASel = IR(9,10);

MuxCSel = 1;

FunSel(ALU) = 0000;

OutDSel = 01;

Load = 1;

Read = 0;

**Opcode2:** DESTREG ← SRCREG1

**IR7 = 1, IR10 = 1**

Opcode2T4: **ARF 🡨 ARF**

OutCSel = SRCREG1-2B-2[2][1]

MUXCSEL = 0

FUNSEL(ALU) = 0000

MUXBSEL=11

FUNSEL(2B) = 01

REGSEL(2B) = DESTREG-2B-2

**IR7 = 1, IR10 = 0**

Opcode2T4: **ARF 🡨 RF**

OutCSel = SRCREG1-2B-2[2][1]

MuxASel=10

FunSel(2A) = 01

RegSel(2A) = DESTREG-2A-2

**IR7 = 0, IR10 = 1**

Opcode2T4: **RF 🡨 ARF**

OutBSel=SRCREG1[1][0];

FunSel(ALU) = 0001;

MuxBSel = 11;

FunSel(2B)=01;

RegSel(2B) = DESTREG-2B-2;

**IR7 = 0, IR10 = 0**

Opcode2T4: **RF 🡨 RF**

OutBSel=SRCREG1[1][0];

FunSel(ALU) = 0001;

MuxBSel = 11;

FunSel(2B)=01;

RegSel(2B) = DESTREG-2B-2;

MuxASel=11;

FunSel(2A)=01;

RegSel(2A) = DESTREG-2A-2;

**Opcode3:** M[SP] ← Rx, SP ← SP - 1

Opcode4T4: **M[SP] 🡨 Rx**

Load = 1;

Read = 0;

OutBSel =  REGSEL-2A-0;

FunSel(ALU)= 0001;

Opcode4T5: **SP 🡨 SP-1**

FunSel(2B) = 11;

RegSel(2B) = 10;

**Opcode4:** SP ← SP + 1, Rx ← M[SP]

Opcode4T4: **SP 🡨 SP + 1**

FunSel(2B) = 10;

RegSel(2B) = 100

Opcode4T5: **Rx 🡨 M[SP]**

OutDSel = 10;

Read = 1;

Load = 0;

MuxASel = 01;

FunSel(2A) = 01;

RegSel(2A) = REGSEL-2A-0

**Opcode5:** DESTREG ← SRCREG1 + SRCREG2

rewind?

**IR10 = 0, IR7=0, IR4 = 0**

Opcode5T4: **RF 🡨 RF + RF**

OutBSel =  IR(2,3)

OutASel = IR(5,6)

MuxCSel = 1;

FunSel(ALU) = 0100

MuxASel = 11;

RegSel =  DESTREG-2A-2

FunSel (2A)= 01

**IR10 = 0, IR7=0, IR4 = 1**

Opcode5T4: **RF 🡨 RF + ARF**

OutCSel = SRCREG1-2B-2[2][1]

MuxCSel = 0;

OutBSel =  IR(2,3);

FunSel(ALU) = 0100

MuxASel = 11;

RegSel =  DESTREG-2A-2

FunSel (2A)= 01

**IR10 = 0, IR7=1, IR4 = 0**

Opcode5T4: **RF 🡨 ARF + RF**

OutCSel = SRCREG1-2B-2[2][1]

MuxCSel = 0;

OutBSel =  IR(5,6);

FunSel(ALU) = 0100

MuxASel = 11;

RegSel =  DESTREG-2A-2

FunSel (2A)= 01

**IR10 = 1, IR7=0, IR4 = 0**

Opcode5T4: **ARF 🡨 RF + RF**

OutBSel =  IR(2,3)

OutASel = IR(5,6)

MuxCSel = 1;

FunSel(ALU) = 0100

MuxBSel = 11

FunSel(2B) = 01

RegSel(2B) = DESTREG-2B-2

**IR10 = 1, IR7=1, IR4 = 0**

Opcode5T4: **ARF 🡨 ARF + RF**

OutCSel = SRCREG1-2B-2[2][1]

MuxCSel = 0

OutBSel = IR(2,3)

FunSel(ALU) = 0100

MuxBSel = 11

FunSel(2B) = 01

RegSel(2B) = DESTREG-2B-2

**IR10 = 1, IR7=0, IR4 = 1**

Opcode5T4: **ARF 🡨 RF + ARF**

OutBSel =  IR(5,6)

OutCSel = SRCREG2-2B-2(1,2)

MuxCSel = 0

FunSelALU = 0100

**Opcode6:** DESTREG ← SRCREG2 - SRCREG1

**IR10 = 0, IR7=0, IR4 = 0**

Opcode5T4: **RF 🡨 RF - RF**

OutASel = SRCREG2(0,1);

MuxCSel = 1

OutBSel = SRCREG1(0,1);

FunSel(ALU) = 0110;

MuxASel = 11;

FunSel(2A) = 01;

RegSel = DESTREG-2A-2

**IR10 = 1, IR7=0, IR4 = 0**

Opcode5T4: **ARF 🡨 RF - RF**

OutASel = SRCREG2(0,1);

MuxCSel = 1

OutBSel =  IR(5,6)

FunSel(ALU) = 0110;

MuxBSel = 11

FunSel(2B) = 01;

RegSel(2B) = DESTREG-2B-2

**IR10 = 0, IR7=0, IR4 = 1**

Opcode5T4: **RF 🡨 ARF – RF**

OutBSel = IR(5,6)

OutCSel ) SRCREG2-2B-2(1,2)

MuxCSel = 0;

FunSel(ALU) = 0110

MuxASel = 11

FunSel(2A) = 01

RegSel(2A) = DESTREG-2A-2

**IR10 = 1, IR7=0, IR4 = 1**

Opcode5T4: **ARF 🡨 ARF - RF**

OutBSel = IR(5,6)

OutCSel ) SRCREG2-2B-2(1,2)

MuxCSel = 0;

FunSel(ALU) = 0110

MuxBSel = 11;

FunSel(2B) = 01;

RegSel(2B) = DESTREG-2B-2

**Opcode7:** DESTREG ← SRCREG1 - 1

**IR10 = 1, IR7 = 1**

Opcode7T4: **ARF 🡨 ARF**

OutCSel = SRCREG-2B-2(1,2)

MuxASel = 10

MuxCSel = 0

FunSel(ALU) = 0000

MuxBSel=11

FUNSEL(2B) = 01

REGSEL(2B) = DESTREG-2B-2

Opcode7T5: **ARF 🡨 ARF - 1**

RegSel(2B) = DESTREG-2B-2

FunSel(2B) = 11

Opcode7T6: **PASSALU ARF**

OutCSel = DESTREG-2B-2(1,2)

MuxCSel = 0;

FunSel(ALU) = 0000;

**IR10 = 0, IR7 = 1:**

Opcode7T4: **RF 🡨 ARF**

OutCSel = SRCREG-2B-2[2][1]

MuxASel=10

FunSel(2A) = 01

RegSel(2A) = DESTREG-2A-2

Opcode7T5: **RF 🡨 RF - 1**

REGSEL = DESTREG-2A-2

FUNSEL(2A) = 11

Opcode7T6: **PASS ALU RF**

OutASel = IR(8,9)

MuxCSel = 1;

FunSel(ALU) = 0000;

**IR10 = 1, IR7 = 0:**

Opcode7T4: **ARF 🡨 RF**

OutBSel=SRCREG1[1][0]

FunSel(ALU) = 0001

MUXBSEL = 11

FUNSEL(2B)=01

REGSEL(2B) = DESTREG-2B-2

Opcode7T5: **ARF 🡨 ARF - 1**

RegSel(2B) = DESTREG-2B-2

FunSel(2B) = 11

Opcode7T6: **PASSALU ARF**

OutCSel = DESTREG-2B-2(1,2)

MuxCSel = 0;

FunSel(ALU) = 0000;

**IR10 = 0, IR7 = 0:**

Opcode7T4: **RF 🡨 RF - 1**

OutBSel=SRCREG1[1][0]

FunSel(ALU) = 0001

MuxASel=11

FunSel(2a)=01

RegSel(2A) = DESTREG-2A-2

Opcode7T5: **RF 🡨 RF - 1**

RegSel(2A) = DESTREG-2A-2

FunSel(2A) = 11

Opcode7T6: **PASSALU RF**

OutASel =  IR(8,9)

MuxCSel = 1;

FunSel(ALU) = 0000;

**Opcode8:** DESTREG ← SRCREG1 + 1

**IR10 = 1, IR7 = 1**

Opcode8T4: **ARF 🡨 ARF**

OutCSel = SRCREG-2B-2[2][1]

MuxCSel = 0

FunSel(ALU) = 0000

MuxBSel=11

FUNSEL(2B) = 01

REGSEL(2B) = DESTREG-2B-2

Opcode8T5: **ARF 🡨 ARF + 1**

RegSel(2B) = DESTREG-2B-2

FunSel(2B) = 10

Opcode8T6: **PASSALU ARF**

OutCSel = DESTREG-2B-2(1,2)

MuxCSel = 0;

FunSel(ALU) = 0000;

**IR10 = 0, IR7 = 1:**

Opcode8T4: **RF 🡨 ARF**

OutCSel = SRCREG-2B-2[2][1]

MuxASel=10

FunSel(2A) = 01

RegSel(2A) = DESTREG-2A-2

Opcode8T5: **RF 🡨 RF + 1**

REGSEL = DESTREG-2A-2

FUNSEL(2A) = 10

Opcode8T6: **PASSALU RF**

OutCSel = DESTREG-2B-2(1,2)

MuxCSel = 0;

FunSel(ALU) = 0000;

**IR10 = 1, IR7 = 0:**

Opcode8T4: **ARF 🡨 RF**

OutBSel=SRCREG1[1][0]

FunSel(ALU) = 0001

MUXBSEL = 11

FUNSEL(2B)=01

REGSEL(2B) = DESTREG-2B-2

Opcode8T5: **ARF 🡨 ARF + 1**

RegSel(2B) = DESTREG-2B-2

FunSel(2B) = 10

Opcode8T6: **PASSALU ARF**

OutCSel = DESTREG-2B-2(1,2)

MuxCSel = 0;

FunSel(ALU) = 0000;

**IR10 = 0, IR7 = 0:**

Opcode8T4: **RF 🡨 RF**

OutBSel=SRCREG1[1][0]

FunSel(ALU) = 0001

MuxASel=11

FunSel(2a)=01

RegSel(2A) = DESTREG-2A-2

Opcode8T5: **RF 🡨 RF + 1**

RegSel(2A) = DESTREG-2A-2

FunSel(2A) = 10

Opcode8T6: **PASSALU RF**

OutASel =  IR(8,9)

MuxCSel = 1;

FunSel(ALU) = 0000;

**Opcode9:** DESTREG ← SRCREG1 AND SRCREG2

**IR4=0,IR7=0,IR10=0**

Opcode9T4: **RF 🡨 RF AND RF**

OutASel = SRCREG1(0,1);

MuxCSel = 1;

OutBSel = SRCREG2(0,1)

FunSel(ALU) = 0111;

MuxASel = 11;

FunSel(2a) = 01;

RegSel(2a) = DESTREG-2A-2

**IR4 = 0, IR7=0, IR10 = 1**

Opcode9T4: **ARF 🡨 RF AND RF**

OutASel = SRCREG1(0,1);

MuxCSel = 1;

OutBSel = SRCREG2(0,1)

FunSel(ALU) = 0111;

MuxBSel = 11;

FunSel(2B) = 01;

RegSel(2B) = DESTREG-2B-2

**IR4 = 1, IR7=0, IR10 = 0:**

Opcode9T4: **RF 🡨 ARF AND RF**

OutBSel = SRCREG1(0,1)

OutCSel = SRCREG2-2B-2(2,1)

MuxCSel = 0;

FunSel(ALU) = 0111

MuxASel = 11;

FunSel(2a) = 01;

RegSel(2A) = DESTREG-2A-2

**IR4 = 1, IR7=0, IR10 = 1:**

Opcode9T4: **ARF 🡨 ARF AND ARF**

OutBSel = SRCREG1(0,1)

OutCSel = SRCREG2-2B-2(2,1)

MuxCSel = 0;

FunSel(ALU) = 0111

MuxBSel = 11;

FunSel(2B) = 01;

RegSel(2B) = DESTREG-2B-2

**IR4 = 0, IR7=1, IR10 = 0:**

Opcode9T4: **RF 🡨 RF AND ARF**

OutBSel = SrCREG2(0,1);

OutCSel = SRCREG1-2B-2(1,2)

MuxCSel = 0;

FunSel(ALU) = 0111

MuxASel = 11;

FunSel(2A) = 01;

RegSel(2a) = DESTREG-2A-2;

**IR4 = 0, IR7=1, IR10 = 1:**

Opcode9T4: **ARF 🡨 RF AND ARF**

OutBSel = SRCREG2(0,1);

OutCSel = SRCREG1-2B-2(1,2)

MuxCSel = 0;

FunSel(ALU) = 0111;

 MuxBSel:11

 FunSel(2B):01

 RegSel(2B):DESTREG-2B-2

**Opcode10:** DESTREG ← SRCREG1 OR SRCREG2

**IR4=0,IR7=0,IR10=0**

Opcode10T4: **RF 🡨 RF OR RF**

OutASel = SRCREG1(0,1);

MuxCSel = 1;

OutBSel = SRCREG2(0,1)

FunSel(ALU) = 1000;

MuxASel = 11;

FunSel(2a) = 01;

RegSel(2a) = DESTREG-2A-2

**IR4 = 0, IR7=0, IR10 = 1**

Opcode10T4: **ARF 🡨 RF OR RF**

OutASel = SRCREG1(0,1);

MuxCSel = 1;

OutBSel = SRCREG’(0,1)

FunSel(ALU) = 1000;

MuxBSel = 11;

FunSel(2B) = 01;

RegSel(2B) = DESTREG-2B-2

**IR4 = 1, IR7=0, IR10 = 0:**

Opcode10T4: **RF 🡨 RF OR ARF**

OutBSel = SRCREG1(0,1)

OutCSel = SRCREG2-2B-2(2,1)

MuxCSel = 0;

FunSel(ALU) = 1000

MuxASel = 11;

FunSel(2a) = 01;

RegSel(2A) = DESTREG-2A-2

**IR4 = 1, IR7=0, IR10 = 1:**

Opcode10T4: **ARF 🡨 RF OR ARF**

OutBSel = SRCREG1(0,1)

OutCSel = SRCREG2-2B-2(2,1)

MuxCSel = 0;

FunSel(ALU) = 1000

MuxBSel = 11;

FunSel(2B) = 01;

RegSel(2B) = DESTREG-2B-2

**IR4 = 0, IR7=1, IR10 = 0:**

Opcode10T4: **RF 🡨 ARF OR RF**

OutBSel = SrCREG2(0,1);

OutCSel = SRCREG1-2B-2(1,2)

MuxCSel = 0;

FunSel(ALU) = 1000

MuxASel = 11;

FunSel(2A) = 01;

RegSel(2a) = DESTREG-2A-2;

**IR4 = 0, IR7=1, IR10 = 1:**

Opcode10T4: **ARF 🡨 ARF OR RF**

OutBSel = SrCREG2(0,1);

OutCSel = SRCREG1-2B-2(1,2)

MuxCSel = 0;

FunSel(ALU) = 1000;

 MuxBSel:11

 FunSel(2B):01

 RegSel(2B):DESTREG-2B-2

**Opcode11:** DESTREG ← NOT SRCREG1

**IR7=0,IR10=0**

Opcode11T4: **RF 🡨 NOT RF**

OutASel = SRCREG1(0,1);

MuxCSel = 1;

FunSel(ALU) = 0010;

MuxASel = 11;

RegSel(2A) = DESTREG-2A-2;

FunSel(2A) = 01

**IR7=0,IR10=1**

Opcode11T4: **ARF 🡨 NOT RF**

OutASel = SRCREG1(0,1);

MuxCSel = 1;

FunSel(ALU) = 0010;

MuxBSel = 11;

RegSel(2B) = DESTREG-2B-2;

FunSel(2B) = 10

**IR7=1,IR10=0**

Opcode11T4: **RF 🡨 NOT ARF**

OutCSel = SRCREG1-2B-2

MuxCSel = 0;

FunSel(ALU) = 0010;

MuxASel = 11;

FunSel(2A) = 01;

RegSel(2A) = DESTREG-2A-2

**IR7=1,IR10=1**

Opcode11T4: **ARF 🡨 NOT ARF**

OutCSel = SRCREG1-2B-2

MuxCSel = 0;

FunSel(ALU) = 0010;

MuxBSel= 11;

FunSel(2B) = 01;

RegSel(2B) = DESTREG-2B-2

**Opcode12:** DESTREG ← LSL SRCREG1

**IR7=0,IR10=0**

Opcode12T4: **RF 🡨 LSL RF**

OutASel = SRCREG1(0,1)

MuxCSel = 1;

FunSel(ALU) = 1010;

MuxASel = 11;

FunSel(2A) = 01;

RegSel(2A) = DESTREG-2A-2

**IR7=0,IR10=1**

Opcode12T4: **ARF 🡨 LSL RF**

OutASel = SRCREG1(0,1)

MuxCSel = 1;

FunSel(ALU) = 1010;

MuxBSel= 11;

FunSel(2B) = 01;

RegSel(2B) = DESTREG-2B-2

**IR7=1,IR10=0**

Opcode12T4: **RF 🡨 LSL ARF**

OutCSel = SRCREG1-2B-2;

MuxCSel = 0;

FunSel(ALU) = 1010;

MuxASel = 11;

FunSel(2A) = 01;

RegSel(2A) = DESTREG-2A-2

**IR7=1,IR10=1**

Opcode12T4: **ARF 🡨 LSL ARF**

OutCSel = SRCREG1-2B-2;

MuxCSel = 0;

FunSel(ALU) = 1010;

MuxBSel = 11;

FunSel(2B) = 01;

RegSel(2B) = DESTREG-2B-2

**Opcode13:** DESTREG ← LSR SRCREG1

**IR7=0,IR10=0**

Opcode13T4: **RF 🡨 LSR RF**

OutASel = SRCREG1(0,1)

MuxCSel = 1;

FunSel(ALU) = 1011;

MuxASel = 11;

FunSel(2A) = 01;

RegSel(2A) = DESTREG-2A-2

**IR7=0,IR10=1**

Opcode13T4: **ARF 🡨 LSR RF**

OutASel = SRCREG1(0,1)

MuxCSel = 1;

FunSel(ALU) = 1011;

MuxBSel= 11;

FunSel(2B) = 01;

RegSel(2B) = DESTREG-2B-2

**IR7=1,IR10=0**

Opcode13T4: **RF 🡨 LSR ARF**

OutCSel = SRCREG1-2B-2;

MuxCSel = 0;

FunSel(ALU) = 1011;

MuxASel = 11;

FunSel(2A) = 01;

RegSel(2A) = DESTREG-2A-2

**IR7=1,IR10=1**

Opcode13T4: **ARF 🡨 LSR ARF**

OutCSel = SRCREG1-2B-2;

MuxCSel = 0;

FunSel(ALU) = 1011;

MuxBSel = 11;

FunSel(2B) = 01;

RegSel(2B) = DESTREG-2B-2

**Opcode14:** PC ← Value

Note: Addressing mode IM

Opcode14T4: **PC 🡨 IROut(0,7)**

MuxBSel = 01;

RegSel(2B) = 001;

FunSel = 01;

**Opcode15:** IF Z = 1 THEN PC ← Value

Note: Addressing mode IM

**Z = 1:**

Opcode15T4: **PC 🡨 IROut(0,7)**

MuxBSel = 01;

RegSel(2B) = 001;

FunSel(2B)= 01;

**Opcode16:** IF Z = 0 THEN PC ← Value

Note: Addressing mode IM

**Z = 0:**

Opcode16T4: **PC 🡨 IROut(0,7)**

MuxBSel = 01;

RegSel(2B) = 001;

FunSel(2B)= 01;

**Opcode17:** M[SP] ← PC, SP ← SP-1, PC ← Value

Note: Addressing mode IM

Opcode17T4: **M[SP] 🡨 PC**

Read = 0;

Load = 1;

OutDSel = 10;

OutCSel = 00;

MuxCSel = 0;

FunSel(ALU) = 0000;

Opcode17T5: **SP 🡨 SP-1**

RegSel(2B) = 100;

FunSel(2B) = 11;

Opcode17T6: **PC 🡨 IROut(0,7)**

MuxBSel = 01;

RegSel(2B) = 01;

FunSel(2B) = 01;

**Opcode18:** SP ← SP+1, PC ← M[SP]

Opcode18T4: **SP 🡨 SP+1**

RegSel(2B) = 100;

FunSel(2B) = 10;

Opcode18T5: **PC 🡨 M[SP]**

OutDSel = 10;

Read = 1;

Load = 0

MuxBSel = 10;

RegSel(2B) = 001;

FunSel(2B) = 01;

-------OPERATIONS-------

**Rx 🡨 IROut(0,7)**

1. **Rx 🡨 M[AR]**
2. **M[AR]** 🡨 **Rx**
3. **ARF 🡨 ARF**
4. **ARF 🡨 RF**
5. **RF 🡨 ARF**
6. **RF 🡨 RF**
7. **M[SP]** 🡨 **Rx**
8. **Rx 🡨 M[SP]**
9. **SP** 🡨 **SP-1**
10. **SP 🡨 SP + 1**
11. **RF 🡨 RF + RF**
12. **RF 🡨 RF + ARF**
13. **RF 🡨 ARF + RF**
14. **ARF 🡨 RF + RF**
15. : **ARF 🡨 ARF + RF**
16. **ARF 🡨 RF + ARF**
17. **RF** 🡨 **RF - RF**
18. **ARF** 🡨 **RF - RF**
19. **RF** 🡨 **ARF - RF**
20. **ARF** 🡨 **ARF - RF**
21. **RF 🡨 RF AND RF**
22. **RF** 🡨 **RF AND RF**
23. **ARF** 🡨 **RF AND RF**
24. **RF** 🡨 **ARF AND RF**
25. **ARF** 🡨 **ARF AND ARF**
26. **RF** 🡨 **RF AND ARF**
27. **ARF** 🡨 **RF AND ARF**
28. **RF** 🡨 **RF OR RF**
29. **ARF** 🡨 **RF OR RF**
30. **RF** 🡨 **RF OR ARF**
31. **ARF** 🡨 **RF OR ARF**
32. **RF** 🡨 **ARF OR RF**
33. **ARF** 🡨 **ARF OR RF**
34. **RF 🡨 NOT RF**
35. **ARF** 🡨 **NOT RF**
36. **RF** 🡨 **NOT ARF**
37. **ARF** 🡨 **NOT ARF**
38. **RF** 🡨 **LSL RF**
39. **ARF** 🡨 **LSL RF**
40. **RF** 🡨 **LSL ARF**
41. **ARF** 🡨 **LSL ARF**
42. **RF 🡨 LSR RF**
43. **ARF** 🡨 **LSR RF**
44. **RF** 🡨 **LSR ARF**
45. **ARF** 🡨 **LSR ARF**
46. **PC** 🡨 **IROut(0,7)**
47. **M[SP] 🡨 PC**
48. **PC** 🡨 **M[SP]**
49. **PASSALU ARF**
50. **PASSALU RF**
51. **ARF 🡨 ARF – 1**
52. **RF 🡨 RF -1**
53. **ARF 🡨 ARF +1**
54. **RF 🡨 RF + 1**
55. **IR(8,15) 🡨 M[PC]**
56. **IR(0,7) 🡨 M[PC]**
57. **PC 🡨 PC+1**

Micro instruction = F(6-bit) CD(condition 2-bit) BR(Branch, 2-bit) AD(Address, 8bit)

Roral 18 bit instruction

Brannch:

MAP

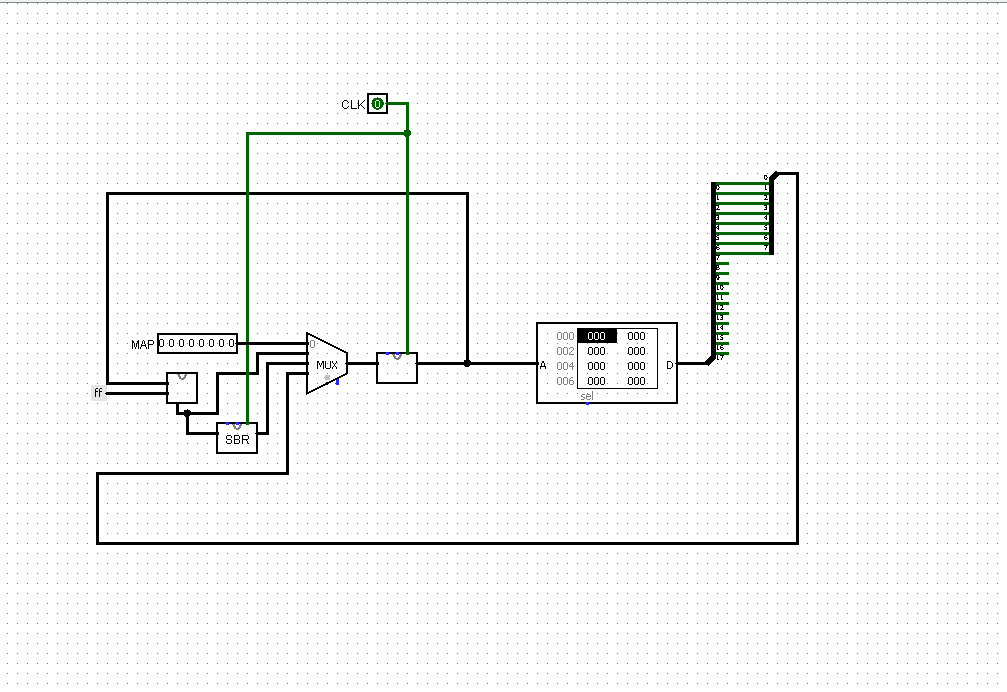
JUMP

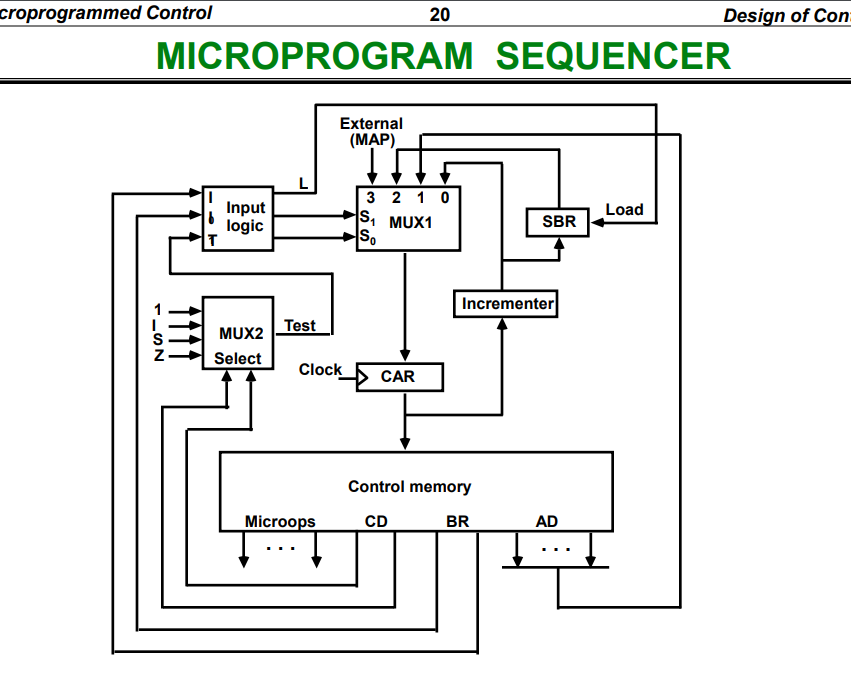
Call

Ret

Rom – 18 bitlik sectionları olacak, address de 8 bitlik olsun

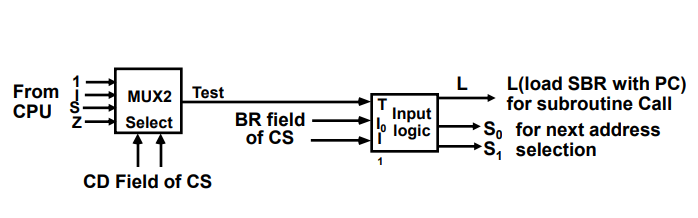
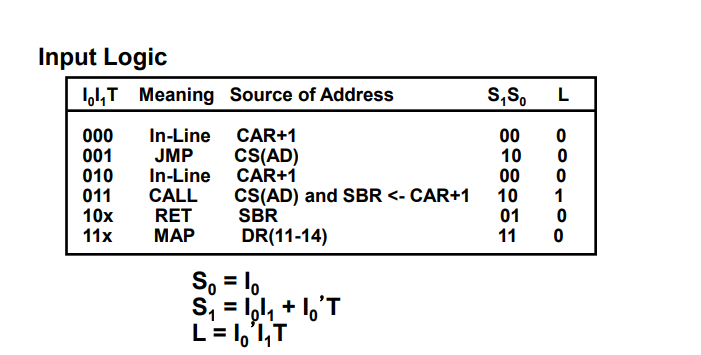
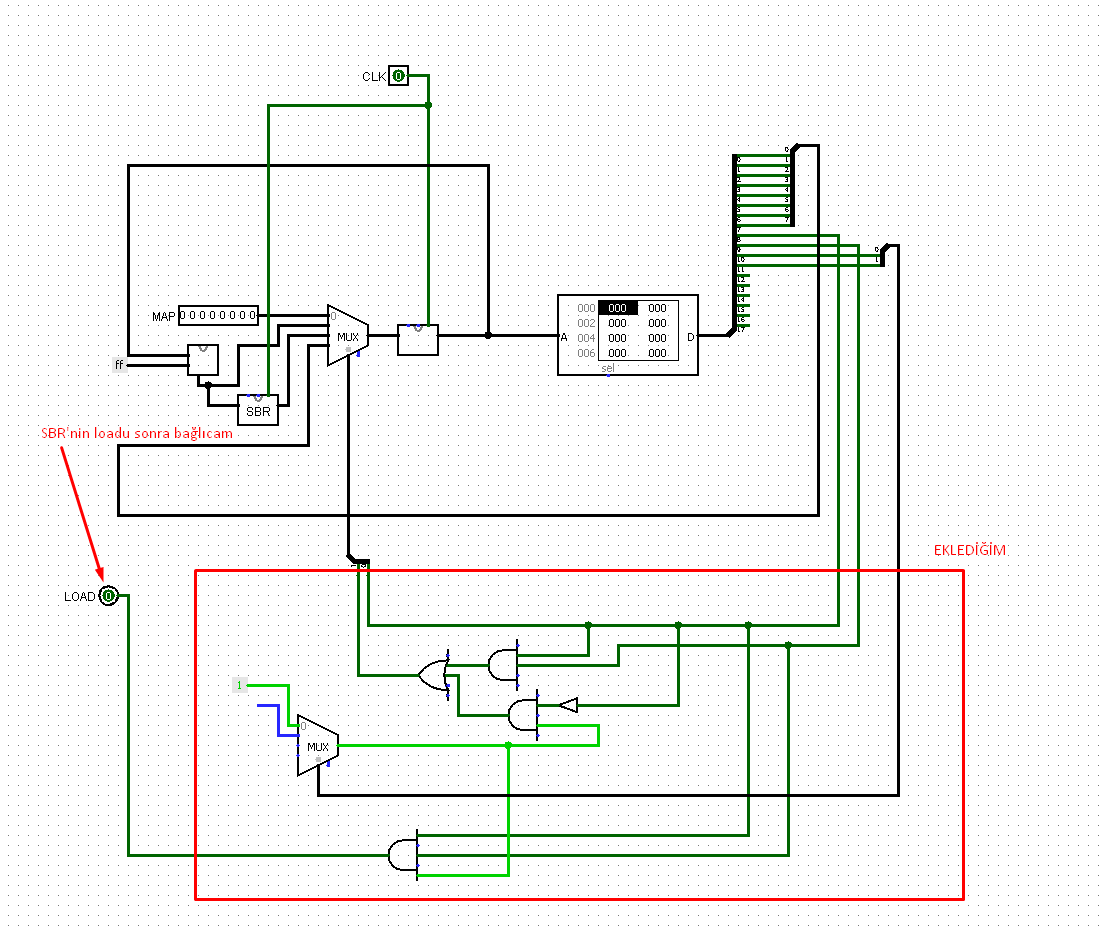
SEQUENCER





SBR nin load dite bir enableı olması lazım. Onu d latche ekleriz ya da clear ile yaparız bilmiyorum şuan

MUX1 de seçmeyi sağlıcak şeyi sonra düşünüp yapıcam.

* CD ve BR yi slaytaki gibi yapıcam.
* 52 micro instruction microops dışındaki kısmını yaparken ona göre yaparız
* 
* 
* 
* MUX1 E GİRENLERİN SIRASI YANLIŞ ŞUAN.
* cd = 00 ise U oluyor, unconditional branch, direk 1

ŞUAN CD LERDEN EMİN DEĞİLİM.

Mesela direk jumpsa branch addresse atlasın call sa call etsin cd ne alaka anlamadım. Z = 0 ise olayın seyri değişiyordu bazı instructionlarda ama. Bu IR8 değerini falan nasıl check edicez?

Maplemede onlara bakmadan yapıyorsak maplemede giden instruction mı bakacak? O nasıl olacak?, CD ler IR4 7 8 10 olsa mesela onun durumuna göre atlasa dicem ad dekine atlar ya da call eder? O kısım nasıl olacak?

Mappingde yapıcaz