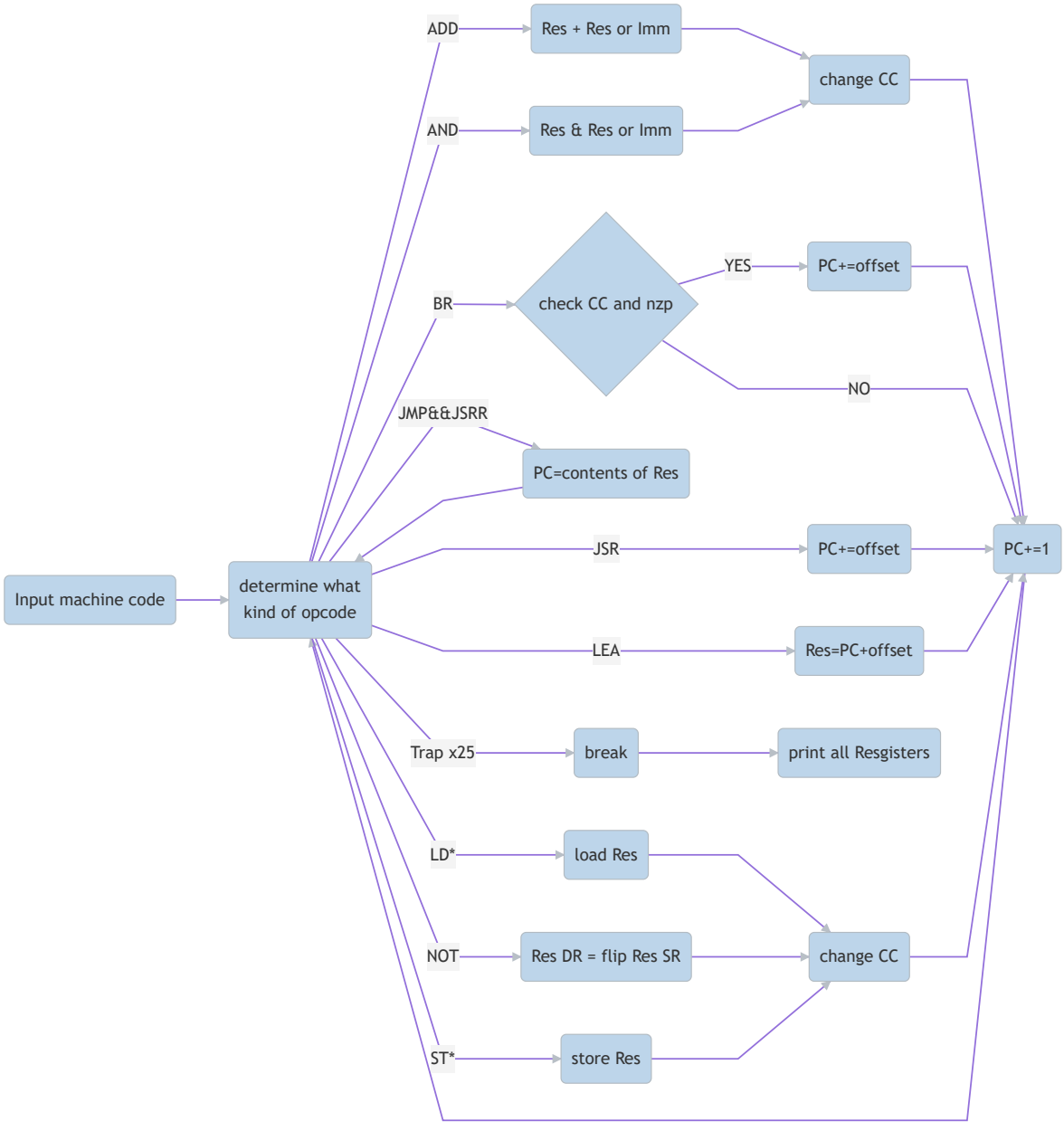
Lab6 Report:

Mermaid:



Opcode ADD:

```
//This part is function ADD
if ( PC[i][0]=='0' && PC[i][1]=='0' && PC[i][2]=='0' && PC[i][3]=='1' ) //ADD
        {
            if (PC[i][10]=='0')//Resgister + Resgister
            {
                int DR,SR1,SR2;
                DR = (PC[i][4]-'0')*4 + (PC[i][5]-'0')*2 + (PC[i][6]-'0');
                SR1 = (PC[i][7]-'0')*4 + (PC[i][8]-'0')*2 + (PC[i][9]-'0');
                SR2 =  (PC[i][13]-'0')*4 + (PC[i][14]-'0')*2 + (PC[i][15]-'0');
                //if Res>0x7fff, Res is negative. if SR1(SR2) is frame pointer or
stack pointer,we shouldn't flip it.
                if ( Res[SR1]>0x7fff && SR1!=6 && SR1!=5)
                {
                    Res[SR1] = ~Res[SR1];
                    Res[SR1] += 1;
                    Res[SR1] = -Res[SR1];
                }
                if ( Res[SR2]>0x7fff && SR2!=6 && SR2!=5)
                {
                    Res[SR2] = ~Res[SR2];
                    Res[SR2] += 1;
                    Res[SR2] = -Res[SR2];
                }
                Res[DR] = Res[SR1] + Res[SR2];
                //set condition code
                if ( Res[DR]>0x7fff )
                    Condition_Code=-1;
                else if (Res[DR]==0x0000)
                    Condition_Code=0;
                else
                    Condition_Code=1;
                i += 1;//PC+=1
                continue;
            }
            else//Resgister + IMM
            {
                int DR,SR1,IMM;
                DR = (PC[i][4]-'0')*4 + (PC[i][5]-'0')*2 + (PC[i][6]-'0');
                SR1 = (PC[i][7]-'0')*4 + (PC[i][8]-'0')*2 + (PC[i][9]-'0');
                if ( PC[i][11]=='0' )//if Imm is positive
                    IMM = (PC[i][12]-'0')*8 + (PC[i][13]-'0')*4 + (PC[i][14]-
'0')*2 + (PC[i][15]-'0');
                else//if Imm is negative
```

```
                {
                    IMM = ('1'-PC[i][12])*8 + ('1'-PC[i][13])*4 + ('1'-PC[i]
[14])*2 + ('1'-PC[i][15]);
                    IMM += 1;
                    IMM = -IMM;
                }
                if ( Res[SR1]>0x7fff && SR1!=6 && SR2!=5)
                {
                    Res[SR1] = ~Res[SR1];
                    Res[SR1] += 1;
                    Res[SR1] = -Res[SR1];
                }
                Res[DR] = Res[SR1] + IMM;
                if ( Res[DR]>0x7fff )
                    Condition_Code=-1;
                else if ( Res[DR]==0x0000 )
                    Condition_Code=0;
                else
                    Condition_Code=1;
                i += 1;
                continue;
            }
        }
```

Opcode AND:

```
//this part is function AND
else if ( PC[i][0]=='0' && PC[i][1]=='1' && PC[i][2]=='0' && PC[i][3]=='1' )
        {
            if ( PC[i][10]=='0' )
            {
                int DR,SR1,SR2;
                DR = (PC[i][4]-'0')*4 + (PC[i][5]-'0')*2 + (PC[i][6]-'0');
                SR1 = (PC[i][7]-'0')*4 + (PC[i][8]-'0')*2 + (PC[i][9]-'0');
                SR2 =  (PC[i][13]-'0')*4 + (PC[i][14]-'0')*2 + (PC[i][15]-'0');
                // & is AND
                Res[DR] = Res[SR1] & Res[SR2];
                //set condition code
                if (Res[DR]>0x7fff)
                    Condition_Code=-1;
                else if (Res[DR]==0x0000)
                    Condition_Code=0;
                else
                    Condition_Code=1;
                i += 1;
                continue;
            }
            else
            {
                int  DR,SR1,IMM;
                DR = (PC[i][4]-'0')*4 + (PC[i][5]-'0')*2 + (PC[i][6]-'0');
                SR1 = (PC[i][7]-'0')*4 + (PC[i][8]-'0')*2 + (PC[i][9]-'0');
                if (PC[i][11]-'0'==0)
                    IMM = (PC[i][12]-'0')*8 + (PC[i][13]-'0')*4 + (PC[i][14]-
'0')*2 + (PC[i][15]-'0');
                else
                {
```

```
                    IMM = ('1'-PC[i][12])*8 + ('1'-PC[i][13])*4 + ('1'-PC[i]
[14])*2 + ('1'-PC[i][15]);
                    IMM += 1;
                    IMM = -IMM;
                }
                Res[DR] = Res[SR1] & IMM;
                if (Res[DR]>0x7fff)
                    Condition_Code=-1;
                else if (Res[DR]==0x0000)
                    Condition_Code=0;
                else
                    Condition_Code=1;
                i += 1;
                continue;
            }
        }
```

Opcode BR:

```
// this part is function BR
else if ( PC[i][0]=='0' && PC[i][1]=='0' && PC[i][2]=='0' && PC[i][3]=='0' )
        {
            int OFFSET=0,j;
            if (PC[i][7]=='0')//if offset is positive
                for (j=8;j<=15;j++)
                    OFFSET = OFFSET*2 + PC[i][j]-'0';
            else//if offset is negative
            {
                for (j=8;j<=15;j++)
                    OFFSET = OFFSET*2 + '1' - PC[i][j];
                OFFSET += 1;
                OFFSET = -OFFSET;
            }
            if ( (Condition_Code==-1 && PC[i][4]=='1') || (Condition_Code==0 &&
PC[i][5]=='1') || (Condition_Code==1 && PC[i][6]=='1') )
                i += OFFSET;//if match successfully, i+=offset;
            i += 1;
            continue;
        }
```

Opcode JMP:

```
// this part is function JMP
else if ( PC[i][0]=='1' && PC[i][1]=='1' && PC[i][2]=='0' && PC[i][3]=='0' )
        {
            int BaseR;
            BaseR = (PC[i][7]-'0')*4 + (PC[i][8]-'0')*2 + (PC[i][9]-'0');
            i = Res[BaseR] - Start_Address;
            i += 1 ;
            continue;
        }
```

Opcode JSR(JSRR):

```
// this part is function JSR and JSRR
else if ( PC[i][0]=='0' && PC[i][1]=='1' && PC[i][2]=='0' && PC[i][3]=='0' )
//JSR
        {
            if (PC[i][4]=='1')
            {
                int j,OFFSET=0;
                Res[7] = i + Start_Address;//store R7
                if (PC[i][5]=='0')
                    for (j=6;j<=15;j++)
                        OFFSET = OFFSET*2 + PC[i][j]-'0';
                else
                {
                    for (j=6;j<=15;j++)
                        OFFSET = OFFSET*2 + '1' - PC[i][j];
                    OFFSET += 1;
                    OFFSET = -OFFSET;
                }
                i += OFFSET + 1;
                continue;
            }
            else //JSRR
            {
                int BaseR;
                Res[7] = i + Start_Address;
                BaseR = (PC[i][7]-'0')*4 + (PC[i][8]-'0')*2 + (PC[i][9]-'0');
                i = Res[BaseR] - Start_Address;
                i += 1 ;
                continue;
            }
        }
```

Opcode LD*:

```
// this part is function LD*
else if ( PC[i][0]=='0' && PC[i][1]=='0' && PC[i][2]=='1' && PC[i][3]=='0' )//LD
        {
            int DR,OFFSET=0,j,temp_PC=0;
            DR = (PC[i][4]-'0')*4 + (PC[i][5]-'0')*2 + (PC[i][6]-'0');
            if (PC[i][7]=='0')
                for (j=8;j<=15;j++)
                    OFFSET = OFFSET*2 + PC[i][j]-'0';
            else
            {
                for (j=8;j<=15;j++)
                    OFFSET = OFFSET*2 + '1' - PC[i][j];
                OFFSET += 1;
                OFFSET = -OFFSET;
            }
            for (j=0;j<=15;j++)//get decimal contents of PC[i+OFFSET+1]
                temp_PC = 2*temp_PC + PC[i+OFFSET+1][j]-'0';
            Res[DR] = temp_PC;
            if ( Res[DR]>0x7fff )
                Condition_Code=-1;
```

```
            else if (Res[DR]==0x0000)
                Condition_Code=0;
            else
                Condition_Code=1;
            i += 1;
            continue;
        }
else  if ( PC[i][0]=='1' && PC[i][1]=='0' && PC[i][2]=='1' && PC[i][3]=='0' )
//LDI
        {
            int DR,OFFSET=0,temp_PC1=0,temp_PC2=0,j;
            DR = (PC[i][4]-'0')*4 + (PC[i][5]-'0')*2 + (PC[i][6]-'0');
            if ( PC[i][7]=='0' )
                for (j=8;j<=15;j++)
                    OFFSET = OFFSET*2 + PC[i][j]-'0';
            else
            {
                for (j=8;j<=15;j++)
                    OFFSET = OFFSET*2 + '1' - PC[i][j];
                OFFSET += 1;
                OFFSET = -OFFSET;
            }
            for (j=0;j<=15;j++)//get indirect address
                temp_PC1 = 2*temp_PC1 + PC[i+OFFSET+1][j]-'0';
            for (j=0;j<=15;j++)//get decimal contents of PC[temp_PC1-
Start_Address+1]
                temp_PC2 = 2*temp_PC2 + PC[temp_PC1-Start_Address+1][j] - '0';
            Res[DR] = temp_PC2;
            if (Res[DR] >0x7fff)
                Condition_Code=-1;
            else if (Res[DR]==0x0000)
                Condition_Code=0;
            else
                Condition_Code=1;
            i += 1;
            continue;
        }
else if ( PC[i][0]=='0' && PC[i][1]=='1' && PC[i][2]=='1' && PC[i][3]=='0'
)//LDR
        {
            int DR,BaseR,OFFSET=0,j,num=0;
            DR = (PC[i][4]-'0')*4 + (PC[i][5]-'0')*2 + (PC[i][6]-'0');
            BaseR = (PC[i][7]-'0')*4 + (PC[i][8]-'0')*2 + (PC[i][9]-'0');
            if ( PC[i][10]=='0' )
                for (j=11;j<=15;j++)
                    OFFSET = 2*OFFSET + PC[i][j] - '0';
            else
            {
                for (j=11;j<=15;j++)
                    OFFSET = 2*OFFSET - PC[i][j] + '1';
                OFFSET += 1;
                OFFSET = -OFFSET;
            }
            for (j=0;j<=15;j++)
                num = 2*num + PC[Res[BaseR]+OFFSET-Start_Address+1][j] - '0';
            Res[DR] = num;
            if (Res[DR]>0x7fff)
                Condition_Code=-1;
```

```
            else if (Res[DR]==0x0000)
                Condition_Code=0;
            else
                Condition_Code=1;
            i += 1;
            continue;
        }
```

Opcode LEA:

```
// this part is function LEA
else if ( PC[i][0]=='1' && PC[i][1]=='1' && PC[i][2]=='1' && PC[i][3]=='0' )
//LEA
        {
            int DR,OFFSET=0,j;
            DR = (PC[i][4]-'0')*4 + (PC[i][5]-'0')*2 + (PC[i][6]-'0');
            if (PC[i][7]=='0')
                for (j=8;j<=15;j++)
                    OFFSET = 2*OFFSET + PC[i][j]-'0';
            else
            {
                for (j=8;j<=15;j++)
                    OFFSET = 2*OFFSET + '1' - PC[i][j];
                OFFSET += 1;
                OFFSET = -OFFSET;
            }
            Res[DR] = Start_Address + OFFSET + i;
            i += 1;
            continue;
        }
```

Opcode NOT:

```
// this part is function not
else if ( PC[i][0]=='1' && PC[i][1]=='0' && PC[i][2]=='0' && PC[i][3]=='1' )
//NOT
        {
            int DR,SR;
            DR = (PC[i][4]-'0')*4 + (PC[i][5]-'0')*2 + (PC[i][6]-'0');
            SR = (PC[i][7]-'0')*4 + (PC[i][8]-'0')*2 + (PC[i][9]-'0');
            Res[DR] = ~Res[SR];// ~ is NOT
            if (Res[DR]>0x7fff)
                Condition_Code=-1;
            else if (Res[DR]==0x0000)
                Condition_Code=0;
            else
                Condition_Code=1;
            i += 1;
            continue;
        }
```

Opcode ST*:

```
// this part is function ST*
        else if ( PC[i][0]=='0' && PC[i][1]=='0' && PC[i][2]=='1' && PC[i]
[3]=='1' )//ST
        {
            int SR,OFFSET=0,j,num;char temp[17];
            SR = (PC[i][4]-'0')*4 + (PC[i][5]-'0')*2 + (PC[i][6]-'0');
            if ( PC[i][7]=='0' )
                for (j=8;j<=15;j++)
                    OFFSET = OFFSET*2 + PC[i][j]-'0';
            else
            {
                for (j=8;j<=15;j++)
                    OFFSET = 2*OFFSET + '1' -PC[i][j];
                OFFSET += 1;
                OFFSET = -OFFSET;
            }
            num = Res[SR];// get binary of Res[SR]
            for (j=15;j>=0;j--)
            {
                temp[j] = num%2 +'0';
                num /= 2;
            }
            temp[16]='\0';
            if ( Res[SR]>0x7fff )
                 Condition_Code=-1;
            else if (Res[SR]==0x0000)
                Condition_Code=0;
            else
                Condition_Code=1;
            strcpy(PC[i+OFFSET+1],temp);
            i += 1;
            continue;
        }
        else if ( PC[i][0]=='1' && PC[i][1]=='0' && PC[i][2]=='1' && PC[i]
[3]=='1' )//STI
        {
            int SR,OFFSET=0,j,num,Address=0;char temp[17];
            SR = (PC[i][4]-'0')*4 + (PC[i][5]-'0')*2 + (PC[i][6]-'0');
            if ( PC[i][7]=='0' )
                for (j=8;j<=15;j++)
                    OFFSET = OFFSET*2 + PC[i][j]- '0';
            else
            {
                for (j=8;j<=15;j++)
                    OFFSET = OFFSET*2 + '1' - PC[i][j];
                OFFSET += 1;
                OFFSET = -OFFSET;
            }
            num = Res[SR];
            for (j=15;j>=0;j--)
            {
                temp[j] = num % 2 + '0';
                num /= 2;
            }
            for (j=0;j<=15;j++)//get decimal indirect address
```

```
                Address = Address*2 + PC[i+OFFSET+1][j]-'0';
            temp[16]='\0';
            strcpy(PC[Address-Start_Address+1],temp);
            if (Res[SR]>0x7fff)
                Condition_Code=-1;
            else if (Res[SR]==0x0000)
                Condition_Code=0;
            else
                Condition_Code=1;
            i += 1;
            continue;
        }
        else if ( PC[i][0]=='0' && PC[i][1]=='1' && PC[i][2]=='1' && PC[i]
[3]=='1' )//STR
        {
            int SR,BaseR,OFFSET=0,j,num,Address;char temp[17];
            SR = (PC[i][4]-'0')*4 + (PC[i][5]-'0')*2 + (PC[i][6]-'0');
            BaseR = (PC[i][7]-'0')*4 + (PC[i][8]-'0')*2 + (PC[i][9]-'0');
            if ( PC[i][10]=='0' )
                for (j=11;j<=15;j++)
                    OFFSET = 2*OFFSET + PC[i][j] - '0';
            else
            {
                for (j=11;j<=15;j++)
                    OFFSET = 2*OFFSET - PC[i][j] + '1';
                OFFSET += 1;
                OFFSET = -OFFSET;
            }
            if (Res[SR]>0x7fff)
                Condition_Code=-1;
            else if (Res[SR]==0)
                Condition_Code=0;
            else
                Condition_Code=1;
            num = Res[SR];
            for (j=15;j>=0;j--)
            {
                temp[j] = num % 2 + '0';
                num /= 2;
            }
            temp[16]='\0';
            Address = Res[BaseR] + OFFSET;
            strcpy(PC[Address-Start_Address+1],temp);
            i += 1;
            continue;
        }
```

Opcode TRAP:

```
else if ( PC[i][0]=='1' && PC[i][1]=='1' && PC[i][2]=='1' && PC[i][3]=='1' )
        {
            break;
        }
```