```
#include "CPU.h"

Input* input;

CPU::CPU(MMU* mmu)
{
    _mmu = mmu;
    input = new Input();
    Reset();

}

CPU::~CPU()
{
}

void CPU::Reset()
{
    _clock.Cycle(0, 0);
    _registers.tClock.Cycle(0, 0);

    _registers.A = 0;
    _registers.F = 0;
    _registers.B = 0;
    _registers.C = 0;
    _registers.D = 0;
    _registers.E = 0;
    _registers.H = 0;
    _registers.L = 0;
    _registers.SP = 0xCFFF;
    _registers.PC = 0x0100;

    for (uint8 i = 0; i < 0xFF; i++)
    {
        _opcodeTable[i] = &CPU::NOP;
    }

    _opcodeTable[0x01] = &CPU::LD_BC_RR;        //0x01 LD BC,d16
    _opcodeTable[0x02] = &CPU::LD_BC_A;     //0x02 LD (BC),A
    _opcodeTable[0x03] = &CPU::INC_BC;      //0x03 INC BC
    _opcodeTable[0x04] = &CPU::INC_B;       //0x04 INC B
    _opcodeTable[0x05] = &CPU::DEC_B;       //0x05 DEC B
    _opcodeTable[0x06] = &CPU::LD_B_R;      //0x06 LD B,d8
    _opcodeTable[0x07] = &CPU::RL_CA;       //0x07
    _opcodeTable[0x08] = &CPU::LD_RR_SP;        //0x08
    _opcodeTable[0x09] = &CPU::ADD_HL_BC;   //0x09
    _opcodeTable[0x0A] = &CPU::LD_A_BC;     //0x0A
    _opcodeTable[0x0B] = &CPU::DEC_BC;      //0x0B
    _opcodeTable[0x0C] = &CPU::INC_C;       //0x0C
    _opcodeTable[0x0D] = &CPU::DEC_C;       //0x0D
    _opcodeTable[0x0E] = &CPU::LD_C_R;      //0x0E
    _opcodeTable[0x0F] = &CPU::RR_CA;       //0x0F

    _opcodeTable[0x10] = &CPU::STOP;            //0x10
    _opcodeTable[0x11] = &CPU::LD_DE_RR;        //0x11
    _opcodeTable[0x12] = &CPU::LD_DE_A;     //0x12
    _opcodeTable[0x13] = &CPU::INC_DE;      //0x13
    _opcodeTable[0x14] = &CPU::INC_D;       //0x14
    _opcodeTable[0x15] = &CPU::DEC_D;       //0x15
    _opcodeTable[0x16] = &CPU::LD_D_R;      //0x16
    _opcodeTable[0x17] = &CPU::RL_A;            //0x17
    _opcodeTable[0x18] = &CPU::JR_R;            //0x18
    _opcodeTable[0x19] = &CPU::ADD_HL_DE;   //0x19
    _opcodeTable[0x1A] = &CPU::LD_A_DE;     //0x1A
    _opcodeTable[0x1B] = &CPU::DEC_DE;      //0x1B
    _opcodeTable[0x1C] = &CPU::INC_E;       //0x1C
    _opcodeTable[0x1D] = &CPU::DEC_E;       //0x1D
    _opcodeTable[0x1E] = &CPU::LD_E_R;      //0x1E
    _opcodeTable[0x1F] = &CPU::RR_A;            //0x1F

    _opcodeTable[0x20] = &CPU::JR_NZ_R;     //0x20
    _opcodeTable[0x21] = &CPU::LD_HL_RR;        //0x21
    _opcodeTable[0x22] = &CPU::LD_MM_PLUS_A; //0x22
    _opcodeTable[0x23] = &CPU::INC_HL;      //0x23
    _opcodeTable[0x24] = &CPU::INC_H;       //0x24
    _opcodeTable[0x25] = &CPU::DEC_H;       //0x25
    _opcodeTable[0x26] = &CPU::LD_H_R;      //0x26
    _opcodeTable[0x27] = &CPU::DA_A;            //0x27
    _opcodeTable[0x28] = &CPU::JR_Z_R;      //0x28
    _opcodeTable[0x29] = &CPU::ADD_HL_HL;   //0x29
    _opcodeTable[0x2A] = &CPU::LD_A_MM_PLUS; //0x2A
    _opcodeTable[0x2B] = &CPU::DEC_HL;      //0x2B
    _opcodeTable[0x2C] = &CPU::INC_L;       //0x2C
    _opcodeTable[0x2D] = &CPU::DEC_L;       //0x2D
    _opcodeTable[0x2E] = &CPU::LD_L_R;      //0x2E
    _opcodeTable[0x2F] = &CPU::CPL;         //0x2F

    _opcodeTable[0x30] = &CPU::JR_NC_R;     //0x30
    _opcodeTable[0x31] = &CPU::LD_SP_RR;        //0x31
    _opcodeTable[0x32] = &CPU::LD_MM_MIN_A; //0x32
    _opcodeTable[0x33] = &CPU::INC_SP;      //0x33
    _opcodeTable[0x34] = &CPU::INC_MM;      //0x34
    _opcodeTable[0x35] = &CPU::DEC_MM;      //0x35
    _opcodeTable[0x36] = &CPU::LD_MM_R;     //0x36
    _opcodeTable[0x37] = &CPU::S_CF;            //0x37
    _opcodeTable[0x38] = &CPU::JR_C_R;      //0x38
    _opcodeTable[0x39] = &CPU::ADD_HL_SP;   //0x39
    _opcodeTable[0x3A] = &CPU::LD_A_MM_MIN; //0x3A
    _opcodeTable[0x3B] = &CPU::DEC_SP;      //0x3B
    _opcodeTable[0x3C] = &CPU::INC_A;       //0x3C
    _opcodeTable[0x3D] = &CPU::DEC_A;       //0x3D
    _opcodeTable[0x3E] = &CPU::LD_A_R;      //0x3E
    _opcodeTable[0x3F] = &CPU::C_CF;            //0x3F

    _opcodeTable[0x40] = &CPU::LD_B_B;      //0x40
    _opcodeTable[0x41] = &CPU::LD_B_C;      //0x41
    _opcodeTable[0x42] = &CPU::LD_B_D;      //0x42
    _opcodeTable[0x43] = &CPU::LD_B_E;      //0x43
    _opcodeTable[0x44] = &CPU::LD_B_H;      //0x44
    _opcodeTable[0x45] = &CPU::LD_B_L;      //0x45
    _opcodeTable[0x46] = &CPU::LD_B_MM;     //0x46
    _opcodeTable[0x47] = &CPU::LD_B_A;      //0x47
    _opcodeTable[0x48] = &CPU::LD_C_B;      //0x48
    _opcodeTable[0x49] = &CPU::LD_C_C;      //0x49
    _opcodeTable[0x4A] = &CPU::LD_C_D;      //0x4A
    _opcodeTable[0x4B] = &CPU::LD_C_E;      //0x4B
    _opcodeTable[0x4C] = &CPU::LD_C_H;      //0x4C
    _opcodeTable[0x4D] = &CPU::LD_C_L;      //0x4D
    _opcodeTable[0x4E] = &CPU::LD_C_MM;     //0x4E
    _opcodeTable[0x4F] = &CPU::LD_C_A;      //0x4F

    _opcodeTable[0x50] = &CPU::LD_D_B;      //0x50
    _opcodeTable[0x51] = &CPU::LD_D_C;      //0x51
    _opcodeTable[0x52] = &CPU::LD_D_D;      //0x52
    _opcodeTable[0x53] = &CPU::LD_D_E;      //0x53
    _opcodeTable[0x54] = &CPU::LD_D_H;      //0x54
    _opcodeTable[0x55] = &CPU::LD_D_L;      //0x55
    _opcodeTable[0x56] = &CPU::LD_D_MM;     //0x56
    _opcodeTable[0x57] = &CPU::LD_D_A;      //0x57
    _opcodeTable[0x58] = &CPU::LD_E_B;      //0x58
    _opcodeTable[0x59] = &CPU::LD_E_C;      //0x59
    _opcodeTable[0x5A] = &CPU::LD_E_D;      //0x5A
    _opcodeTable[0x5B] = &CPU::LD_E_E;      //0x5B
    _opcodeTable[0x5C] = &CPU::LD_E_H;      //0x5C
    _opcodeTable[0x5D] = &CPU::LD_E_L;      //0x5D
    _opcodeTable[0x5E] = &CPU::LD_E_MM;     //0x5E
    _opcodeTable[0x5F] = &CPU::LD_E_A;      //0x5F

    _opcodeTable[0x60] = &CPU::LD_H_B;      //0x60
    _opcodeTable[0x61] = &CPU::LD_H_C;      //0x61
    _opcodeTable[0x62] = &CPU::LD_H_D;      //0x62
    _opcodeTable[0x63] = &CPU::LD_H_E;      //0x63
    _opcodeTable[0x64] = &CPU::LD_H_H;      //0x64
    _opcodeTable[0x65] = &CPU::LD_H_L;      //0x65
    _opcodeTable[0x66] = &CPU::LD_H_MM;     //0x66
    _opcodeTable[0x67] = &CPU::LD_H_A;      //0x67
    _opcodeTable[0x68] = &CPU::LD_L_B;      //0x68
    _opcodeTable[0x69] = &CPU::LD_L_C;      //0x69
    _opcodeTable[0x6A] = &CPU::LD_L_D;      //0x6A
    _opcodeTable[0x6B] = &CPU::LD_L_E;      //0x6B
    _opcodeTable[0x6C] = &CPU::LD_L_H;      //0x6C
    _opcodeTable[0x6D] = &CPU::LD_L_L;      //0x6D
    _opcodeTable[0x6E] = &CPU::LD_L_MM;     //0x6E
    _opcodeTable[0x6F] = &CPU::LD_L_A;      //0x6F

    _opcodeTable[0x70] = &CPU::LD_MM_B;     //0x70
    _opcodeTable[0x71] = &CPU::LD_MM_C;     //0x71
    _opcodeTable[0x72] = &CPU::LD_MM_D;     //0x72
    _opcodeTable[0x73] = &CPU::LD_MM_E;     //0x73
    _opcodeTable[0x74] = &CPU::LD_MM_H;     //0x74
    _opcodeTable[0x75] = &CPU::LD_MM_L;     //0x75
    _opcodeTable[0x76] = &CPU::HALT;            //0x76
    _opcodeTable[0x77] = &CPU::LD_MM_A;     //0x77
    _opcodeTable[0x78] = &CPU::LD_A_B;      //0x78
    _opcodeTable[0x79] = &CPU::LD_A_C;      //0x79
    _opcodeTable[0x7A] = &CPU::LD_A_D;      //0x7A
    _opcodeTable[0x7B] = &CPU::LD_A_E;      //0x7B
    _opcodeTable[0x7C] = &CPU::LD_A_H;      //0x7C
    _opcodeTable[0x7D] = &CPU::LD_A_L;      //0x7D
    _opcodeTable[0x7E] = &CPU::LD_A_MM;     //0x7E
    _opcodeTable[0x7F] = &CPU::LD_A_A;      //0x7F


    _opcodeTable[0x80] = &CPU::ADD_A_B;     //0x80
    _opcodeTable[0x81] = &CPU::ADD_A_C;     //0x81
    _opcodeTable[0x82] = &CPU::ADD_A_D;     //0x82
    _opcodeTable[0x83] = &CPU::ADD_A_E;     //0x83
    _opcodeTable[0x84] = &CPU::ADD_A_H;     //0x84
    _opcodeTable[0x85] = &CPU::ADD_A_L;     //0x85
    _opcodeTable[0x86] = &CPU::ADD_A_MM;        //0x86
    _opcodeTable[0x87] = &CPU::ADD_A_A;     //0x87
    _opcodeTable[0x88] = &CPU::ADC_A_B;     //0x88
    _opcodeTable[0x89] = &CPU::ADC_A_C;     //0x89
    _opcodeTable[0x8A] = &CPU::ADC_A_D;     //0x8A
    _opcodeTable[0x8B] = &CPU::ADC_A_E;     //0x8B
    _opcodeTable[0x8C] = &CPU::ADC_A_H;     //0x8C
    _opcodeTable[0x8D] = &CPU::ADC_A_L;     //0x8D
    _opcodeTable[0x8E] = &CPU::ADC_A_MM;        //0x8E
    _opcodeTable[0x8F] = &CPU::ADC_A_A;     //0x8F

    _opcodeTable[0x90] = &CPU::SUB_A_B;     //0x90
    _opcodeTable[0x91] = &CPU::SUB_A_C;     //0x91
    _opcodeTable[0x92] = &CPU::SUB_A_D;     //0x92
    _opcodeTable[0x93] = &CPU::SUB_A_E;     //0x93
    _opcodeTable[0x94] = &CPU::SUB_A_H;     //0x94
    _opcodeTable[0x95] = &CPU::SUB_A_L;     //0x95
    _opcodeTable[0x96] = &CPU::SUB_A_MM;        //0x96
    _opcodeTable[0x97] = &CPU::SUB_A_A;     //0x97
    _opcodeTable[0x98] = &CPU::SBC_A_B;     //0x98
    _opcodeTable[0x99] = &CPU::SBC_A_C;     //0x99
    _opcodeTable[0x9A] = &CPU::SBC_A_D;     //0x9A
    _opcodeTable[0x9B] = &CPU::SBC_A_E;     //0x9B
    _opcodeTable[0x9C] = &CPU::SBC_A_H;     //0x9C
    _opcodeTable[0x9D] = &CPU::SBC_A_L;     //0x9D
    _opcodeTable[0x9E] = &CPU::SBC_A_MM;        //0x9E
    _opcodeTable[0x9F] = &CPU::SBC_A_A;     //0x9F

    _opcodeTable[0xA0] = &CPU::AND_B;       //0xA0
    _opcodeTable[0xA1] = &CPU::AND_C;       //0xA1
    _opcodeTable[0xA2] = &CPU::AND_D;       //0xA2
    _opcodeTable[0xA3] = &CPU::AND_E;       //0xA3
    _opcodeTable[0xA4] = &CPU::AND_H;       //0xA4
    _opcodeTable[0xA5] = &CPU::AND_L;       //0xA5
    _opcodeTable[0xA6] = &CPU::AND_MM;      //0xA6
    _opcodeTable[0xA7] = &CPU::AND_A;       //0xA7
    _opcodeTable[0xA8] = &CPU::XOR_B;       //0xA8
    _opcodeTable[0xA9] = &CPU::XOR_C;       //0xA9
    _opcodeTable[0xAA] = &CPU::XOR_D;       //0xAA
    _opcodeTable[0xAB] = &CPU::XOR_E;       //0xAB
    _opcodeTable[0xAC] = &CPU::XOR_H;       //0xAC
    _opcodeTable[0xAD] = &CPU::XOR_L;       //0xAD
    _opcodeTable[0xAE] = &CPU::XOR_MM;      //0xAE
    _opcodeTable[0xAF] = &CPU::XOR_A;       //0xAF

    _opcodeTable[0xB0] = &CPU::OR_B;            //0xB0
    _opcodeTable[0xB1] = &CPU::OR_C;            //0xB1
    _opcodeTable[0xB2] = &CPU::OR_D;            //0xB2
    _opcodeTable[0xB3] = &CPU::OR_E;            //0xB3
    _opcodeTable[0xB4] = &CPU::OR_H;            //0xB4
    _opcodeTable[0xB5] = &CPU::OR_L;            //0xB5
    _opcodeTable[0xB6] = &CPU::OR_MM;       //0xB6
    _opcodeTable[0xB7] = &CPU::OR_A;            //0xB7
    _opcodeTable[0xB8] = &CPU::CP_B;            //0xB8
    _opcodeTable[0xB9] = &CPU::CP_C;            //0xB9
    _opcodeTable[0xBA] = &CPU::CP_D;            //0xBA
    _opcodeTable[0xBB] = &CPU::CP_E;            //0xBB
    _opcodeTable[0xBC] = &CPU::CP_H;            //0xBC
    _opcodeTable[0xBD] = &CPU::CP_L;            //0xBD
    _opcodeTable[0xBE] = &CPU::CP_MM;       //0xBE
    _opcodeTable[0xBF] = &CPU::CP_A;            //0xBF

    _opcodeTable[0xC0] = &CPU::RET_NZ;      //0xC0
    _opcodeTable[0xC1] = &CPU::POP_BC;      //0xC1
    _opcodeTable[0xC2] = &CPU::JP_NZ_RR;        //0xC2
    _opcodeTable[0xC3] = &CPU::JP_RR;       //0xC3
    _opcodeTable[0xC4] = &CPU::CALL_NZ_RR;  //0xC4
    _opcodeTable[0xC5] = &CPU::PUSH_BC;     //0xC5
    _opcodeTable[0xC6] = &CPU::ADD_A_R;     //0xC6
    _opcodeTable[0xC7] = &CPU::RST_00H;     //0xC7
    _opcodeTable[0xC8] = &CPU::RET_Z;       //0xC8
    _opcodeTable[0xC9] = &CPU::RET;         //0xC9
    _opcodeTable[0xCA] = &CPU::JP_Z_RR;     //0xCA
    _opcodeTable[0xCB] = &CPU::PREFIX_CB;   //0xCB
    _opcodeTable[0xCC] = &CPU::CALL_Z_RR;   //0xCC
    _opcodeTable[0xCD] = &CPU::CALL_RR;     //0xCD
    _opcodeTable[0xCE] = &CPU::ADC_A_R;     //0xCE
    _opcodeTable[0xCF] = &CPU::RST_08H;     //0xCF

    _opcodeTable[0xD0] = &CPU::RET_NC;      //0xD0
    _opcodeTable[0xD1] = &CPU::POP_DE;      //0xD1
    _opcodeTable[0xD2] = &CPU::JP_NC_RR;        //0xD2
    _opcodeTable[0xD4] = &CPU::CALL_NC_RR;  //0xD4
    _opcodeTable[0xD5] = &CPU::PUSH_DE;     //0xD5
    _opcodeTable[0xD6] = &CPU::SUB_R;       //0xD6
    _opcodeTable[0xD7] = &CPU::RST_10H;     //0xD7
    _opcodeTable[0xD8] = &CPU::RET_C;       //0xD8
    _opcodeTable[0xD9] = &CPU::RETI;            //0xD9
    _opcodeTable[0xDA] = &CPU::JP_C_RR;     //0xDA
    _opcodeTable[0xDC] = &CPU::CALL_C_RR;   //0xDC
    _opcodeTable[0xDE] = &CPU::SBC_A_R;     //0xDE
    _opcodeTable[0xDF] = &CPU::RST_18H;     //0xDF

    _opcodeTable[0xE0] = &CPU::LDH_R_A;     //0xE0
    _opcodeTable[0xE1] = &CPU::POP_HL;      //0xE1
    _opcodeTable[0xE2] = &CPU::LD_C_A2;     //0xE2
    _opcodeTable[0xE5] = &CPU::PUSH_HL;     //0xE5
    _opcodeTable[0xE6] = &CPU::AND_R;       //0xE6
    _opcodeTable[0xE7] = &CPU::RST_20H;     //0xE7
    _opcodeTable[0xE8] = &CPU::ADD_SP_R;        //0xE8
    _opcodeTable[0xE9] = &CPU::JP_MM;       //0xE9
    _opcodeTable[0xEA] = &CPU::LD_RR_A;     //0xEA
    _opcodeTable[0xEE] = &CPU::XOR_R;       //0xEE
    _opcodeTable[0xEF] = &CPU::RST_28H;     //0xEF

    _opcodeTable[0xF0] = &CPU::LDH_A_R;     //0xF0
    _opcodeTable[0xF1] = &CPU::POP_AF;      //0xF1
    _opcodeTable[0xF2] = &CPU::LD_A_C2;     //0xF2
    _opcodeTable[0xF3] = &CPU::DI;          //0xF3
    _opcodeTable[0xF5] = &CPU::PUSH_AF;     //0xF5
    _opcodeTable[0xF6] = &CPU::OR_R;            //0xF6
    _opcodeTable[0xF7] = &CPU::RST_30H;     //0xF7
    _opcodeTable[0xF8] = &CPU::LD_HL_SPandR;    //0xF8
    _opcodeTable[0xF9] = &CPU::LD_SP_HL;        //0xF9
    _opcodeTable[0xFA] = &CPU::LD_A_RR;     //0xFA
    _opcodeTable[0xFB] = &CPU::EI;          //0xFB
    _opcodeTable[0xFE] = &CPU::CP_R;            //0xFE
    _opcodeTable[0xFF] = &CPU::RST_38H;     //0xFF
}

bool CPU::Run()
{
    //(expected sequence)
    BOOL runningState = TRUE;

    /*
    NOP, JP_MM($0150), JP_MM($0185), LD_A_R(A=($180)=0x03),
    DI, LDH_R_A($FF0F <- A), LDH_R_A($FFFF <- A), LD_A_R(a <- 0x40),
    LDH_R_A($FF41 <- 0x40), XOR_A(A <- 0), LDH_R_A($FF42 <- 0x00),
    LDH_R_A($FF43 <- 0x00),LDH_R_A($FF44 <- 0x00),
    */
    uint8 pc = Read8(_registers.PC);
    _registers.PC += 0x0001;
    if (pc == 0xCB)
    {
        //cb
        uint8 pc = Read8(_registers.PC + 1);
        _cbOpcodeTable[pc];
    }
    else
    {
        (this->*(_opcodeTable[pc]))();
    }

    pc += _registers.tClock.byte_call_cycles;
    if (_hasSetPC == FALSE)
    {
        _registers.PC = pc;
    }

    //reset
    _registers.tClock.Cycle(0, 0);
    _hasSetPC = FALSE;

    if (input->IsExit()) { runningState = FALSE; }
    return runningState;
}
```

```

uint8 CPU::Read8(uint16 address)
{
    return _mmu->Read8(address);
}

uint16 CPU::Read16(uint16 address)
{
    return _mmu->Read16(address);
}

void CPU::Write8(uint16 address, uint8 value)
{
    return _mmu->Write8(address, value);
}

void CPU::Write16(uint16 address, uint16 value)
{
    return _mmu->Write16(address, value);
}

void CPU::Set_Z(BOOL value)
{
    Set_Bit(_registers.F, 7, value);
}

void CPU::Set_N(BOOL value)
{
    Set_Bit(_registers.F, 6, value);
}

void CPU::Set_H(BOOL value)
{
    Set_Bit(_registers.F, 5, value);
}
void CPU::Set_C(BOOL value)
{
    Set_Bit(_registers.F, 4, value);
}

void CPU::NOP()
{
    _registers.tClock.Cycle(1, 4);
}

void CPU::LD_BC_RR()
{
    Write8(_registers.BC(), Read8(_registers.PC));
    _registers.tClock.Cycle(3, 12);
}

void CPU::LD_BC_A()
{
    Write8(_registers.BC(), _registers.A);
    _registers.tClock.Cycle(1, 8);
}
```

```

void CPU::INC_BC()
{
    uint16 tempBC = _registers.BC();
    tempBC += 0x0001;
    _registers.BC(tempBC);

    _registers.tClock.Cycle(1, 8);
}

void CPU::INC_B()
{
    INC_M(_registers.B);
}

void CPU::DEC_B()
{
    DEC_M(_registers.B);
}

void CPU::LD_B_R()
{
    _registers.B = Read8(_registers.PC);
    _registers.tClock.Cycle(2, 8);
}

void CPU::RL_CA()
{
    uint8 oldA = _registers.A;
    _registers.A = _registers.A << 1;

    Set_Z(_registers.A == 0);
    Set_N(FALSE);
    Set_H(FALSE);
    Set_C(Get_Bit(oldA, 7));

    _registers.tClock.Cycle(1, 4);
}

void CPU::LD_RR_SP()
{
    Write8(Read16(_registers.PC), Read8(_registers.SP));
    _registers.tClock.Cycle(3, 20);
}

void CPU::ADD_HL_BC()
{
    _registers.HL(_registers.HL() + _registers.BC());
    _registers.tClock.Cycle(1, 8);
}

void CPU::LD_A_BC()
{
    _registers.A = Read8(_registers.BC());
    _registers.tClock.Cycle(1, 8);
}
```

```

void CPU::DEC_BC()
{
    uint16 tempBC = _registers.BC();
    tempBC -= 0x0001;
    _registers.BC(tempBC);

    _registers.tClock.Cycle(1, 8);
}

void CPU::INC_C()
{
    INC_M(_registers.C);
}

void CPU::DEC_C()
{
    DEC_M(_registers.C);
}

void CPU::LD_C_R()
{
    _registers.C = Read8(_registers.PC);
    _registers.tClock.Cycle(2, 8);
}

void CPU::RR_CA()
{
    uint8 oldA = _registers.A;
    _registers.A = _registers.A >> 1;

    Set_Z(_registers.A == 0);
    Set_N(FALSE);
    Set_H(FALSE);
    Set_C(Get_Bit(oldA, 0));

    _registers.tClock.Cycle(1, 4);
}

void CPU::STOP()
{
    //TODO: STOP
    _registers.tClock.Cycle(2, 4);
}

void CPU::LD_DE_RR()
{
    uint16 tempDE = _registers.DE();
    tempDE = Read16(_registers.PC);
    _registers.DE(tempDE);
    _registers.tClock.Cycle(3, 12);
}

void CPU::LD_DE_A()
{
    Write8(Read16(_registers.DE()), _registers.A);
    _registers.tClock.Cycle(1, 8);
}

void CPU::INC_DE()
{
    uint16 tempDE = _registers.DE();
    tempDE += 0x0001;
    _registers.DE(tempDE);
    _registers.tClock.Cycle(1, 8);
}

void CPU::INC_D()
{
    _registers.D += 0x01;
    _registers.tClock.Cycle(1, 4);
}

void CPU::DEC_D()
{
    _registers.D -= 0x01;
    _registers.tClock.Cycle(1, 4);
}

void CPU::LD_D_R()
{
    _registers.D = Read8(_registers.PC);
    _registers.tClock.Cycle(2, 8);
}

void CPU::RL_A()
{
    uint8 oldA = _registers.A;
    _registers.A = _registers.A << 1;
    Set_Bit(_registers.A, 0, Get_Bit(_registers.F, 5));
    Set_C(Get_Bit(oldA, 0));
    Set_Z(_registers.A == 0);
    Set_N(0);
    Set_H(0);
    _registers.tClock.Cycle(1, 4);
}

void CPU::JR_R()
{
    _registers.PC += Read8(_registers.PC);
    _hasSetPC = TRUE;
    _registers.tClock.Cycle(2, 12);
}

void CPU::ADD_HL_DE()
{
    uint16 tempHL = _registers.HL();
    tempHL += _registers.DE();
    _registers.HL(tempHL);
    Set_N(FALSE);
    Set_H(Get_Bit(_registers.HL(), 11));

    _registers.tClock.Cycle(1, 8);
}

void CPU::LD_A_DE()
{
    _registers.tClock.Cycle(1, 8);
}

void CPU::DEC_DE()
{
    _registers.tClock.Cycle(1, 8);
}

void CPU::INC_E()
{
    _registers.tClock.Cycle(1, 4);
}

void CPU::DEC_E()
{
    _registers.tClock.Cycle(1, 4);
}

void CPU::LD_E_R()
{
    _registers.tClock.Cycle(2, 8);
}

void CPU::RR_A()
{
    _registers.tClock.Cycle(1, 4);
}

void CPU::JR_NZ_R()
{
    //(depending on result)
    _registers.tClock.Cycle(2, 12);
    _registers.tClock.Cycle(2, 8);
}

void CPU::LD_HL_RR()
{
    _registers.tClock.Cycle(1, 4);
}

void CPU::LD_MM_PLUS_A()
{
    _registers.tClock.Cycle(1, 8);
}

void CPU::INC_HL()
{
    _registers.tClock.Cycle(1, 8);
}

void CPU::INC_H()
```

```
{
    _registers.tClock.Cycle(1, 4);
}

void CPU::DEC_H()
{
    _registers.tClock.Cycle(1, 4);
}

void CPU::LD_H_R()
{
    _registers.tClock.Cycle(2, 8);
}

void CPU::DA_A()
{
    _registers.tClock.Cycle(1, 4);
}

void CPU::JR_Z_R()
{
    //(depending on result)
    _registers.tClock.Cycle(2, 12);
    _registers.tClock.Cycle(2, 8);
}

void CPU::ADD_HL_HL()
{
    _registers.tClock.Cycle(1, 8);
}

void CPU::LD_A_MM_PLUS()
{
    _registers.tClock.Cycle(1, 8);
}

void CPU::DEC_HL()
{
    _registers.tClock.Cycle(1, 8);
}

void CPU::INC_L()
{
    _registers.tClock.Cycle(1, 4);
}

void CPU::DEC_L()
{
    _registers.tClock.Cycle(1, 4);
}

void CPU::LD_L_R()
{
    _registers.tClock.Cycle(2, 8);
}

```

```
void CPU::CPL()
{
    _registers.tClock.Cycle(1, 4);
}

void CPU::JR_NC_R()
{
    //(depending on result)
    _registers.tClock.Cycle(2, 12);
    _registers.tClock.Cycle(2, 8);
}

void CPU::LD_SP_RR()
{
    _registers.tClock.Cycle(3, 12);
}

void CPU::LD_MM_MIN_A()
{
    _registers.tClock.Cycle(1, 8);
}

void CPU::INC_SP()
{
    _registers.tClock.Cycle(1, 8);
}

void CPU::INC_MM()
{
    _registers.tClock.Cycle(1, 12);
}

void CPU::DEC_MM()
{
    _registers.tClock.Cycle(1, 12);
}

void CPU::LD_MM_R()
{
    _registers.tClock.Cycle(2, 12);
}

void CPU::S_CF()
{
    _registers.tClock.Cycle(1, 4);
}

void CPU::JR_C_R()
{
    //(depending on result)
    _registers.tClock.Cycle(2, 12);
    _registers.tClock.Cycle(2, 8);
}

void CPU::ADD_HL_SP()
{
    _registers.tClock.Cycle(1, 8);
}

void CPU::LD_A_MM_MIN()
{
    _registers.A = Read8(_registers.HL() - 0x0001);
    _registers.tClock.Cycle(1, 8);
}

void CPU::DEC_SP()
{
    _registers.SP--;
    _registers.tClock.Cycle(1, 8);
}

void CPU::INC_A()
{
    _registers.A++;
    _registers.tClock.Cycle(1, 4);
}

void CPU::DEC_A()
{
    _registers.A--;
    _registers.tClock.Cycle(1, 4);
}

void CPU::LD_A_R()
{
    _registers.A = Read8(_registers.PC);
    _registers.tClock.Cycle(2, 8);
}

void CPU::C_CF()
{
    _registers.tClock.Cycle(1, 4);
}

void CPU::LD_B_B()
{
    LD_M_M(_registers.B, _registers.B);
}

void CPU::LD_B_C()
{
    LD_M_M(_registers.B, _registers.C);
}

void CPU::LD_B_D()
{
    LD_M_M(_registers.B, _registers.D);
}

void CPU::LD_B_E()
{
    LD_M_M(_registers.B, _registers.E);
```

```
}

void CPU::LD_B_H()
{
    LD_M_M(_registers.B, _registers.H);
}

void CPU::LD_B_L()
{
    LD_M_M(_registers.B, _registers.L);
}

void CPU::LD_B_MM()
{
}

void CPU::LD_B_A()
{
    LD_M_M(_registers.B, _registers.A);
}

void CPU::LD_C_B()
{
    LD_M_M(_registers.C, _registers.B);
}

void CPU::LD_C_C()
{
    LD_M_M(_registers.C, _registers.C);
}

void CPU::LD_C_D()
{
    LD_M_M(_registers.C, _registers.D);
}

void CPU::LD_C_E()
{
    LD_M_M(_registers.C, _registers.E);
}

void CPU::LD_C_H()
{
    LD_M_M(_registers.C, _registers.H);
}

void CPU::LD_C_L()
{
    LD_M_M(_registers.C, _registers.L);
}

void CPU::LD_C_MM()
{
}

void CPU::LD_C_A()
```

```
{
    LD_M_M(_registers.C, _registers.A);
}

void CPU::LD_D_B()
{
    LD_M_M(_registers.D, _registers.B);
}

void CPU::LD_D_C()
{
    LD_M_M(_registers.D, _registers.C);
}

void CPU::LD_D_D()
{
    LD_M_M(_registers.D, _registers.D);
}

void CPU::LD_D_E()
{
    LD_M_M(_registers.D, _registers.E);
}

void CPU::LD_D_H()
{
    LD_M_M(_registers.D, _registers.H);
}

void CPU::LD_D_L()
{
    LD_M_M(_registers.D, _registers.L);
}

void CPU::LD_D_MM()
{
}

void CPU::LD_D_A()
{
    LD_M_M(_registers.D, _registers.A);
}

void CPU::LD_E_B()
{
    LD_M_M(_registers.E, _registers.B);
}

void CPU::LD_E_C()
{
    LD_M_M(_registers.E, _registers.C);
}

void CPU::LD_E_D()
{
    LD_M_M(_registers.E, _registers.D);
}

void CPU::LD_E_E()
{
    LD_M_M(_registers.E, _registers.E);
}

void CPU::LD_E_H()
{
    LD_M_M(_registers.E, _registers.H);
}

void CPU::LD_E_L()
{
    LD_M_M(_registers.E, _registers.L);
}

void CPU::LD_E_MM()
{
}

void CPU::LD_E_A()
{
    LD_M_M(_registers.E, _registers.A);
}

void CPU::LD_H_B()
{
    LD_M_M(_registers.H, _registers.B);
}

void CPU::LD_H_C()
{
    LD_M_M(_registers.H, _registers.C);
}

void CPU::LD_H_D()
{
    LD_M_M(_registers.H, _registers.D);
}

void CPU::LD_H_E()
{
    LD_M_M(_registers.H, _registers.E);
}

void CPU::LD_H_H()
{
    LD_M_M(_registers.H, _registers.H);
}

void CPU::LD_H_L()
{
    LD_M_M(_registers.H, _registers.L);
}

```

```cpp
void CPU::LD_H_MM()
{
}

void CPU::LD_H_A()
{
    LD_M_M(_registers.H, _registers.A);
}

void CPU::LD_L_B()
{
    LD_M_M(_registers.L, _registers.B);
}

void CPU::LD_L_C()
{
    LD_M_M(_registers.L, _registers.C);
}

void CPU::LD_L_D()
{
    LD_M_M(_registers.L, _registers.D);
}

void CPU::LD_L_E()
{
    LD_M_M(_registers.L, _registers.E);
}

void CPU::LD_L_H()
{
    LD_M_M(_registers.L, _registers.H);
}

void CPU::LD_L_L()
{
    LD_M_M(_registers.L, _registers.L);
}

void CPU::LD_L_MM()
{
}

void CPU::LD_L_A()
{
    LD_M_M(_registers.L, _registers.A);
}

void CPU::LD_MM_B()
{
}

void CPU::LD_MM_C()
{
}

```

```
void CPU::LD_MM_D()
{
}

void CPU::LD_MM_E()
{
}

void CPU::LD_MM_H()
{
}

void CPU::LD_MM_L()
{
}

void CPU::HALT()
{
}

void CPU::LD_MM_A()
{
}

void CPU::LD_A_B()
{
    LD_M_M(_registers.A, _registers.B);
}

void CPU::LD_A_C()
{
    LD_M_M(_registers.A, _registers.C);
}

void CPU::LD_A_D()
{
    LD_M_M(_registers.A, _registers.D);
}

void CPU::LD_A_E()
{
    LD_M_M(_registers.A, _registers.E);
}

void CPU::LD_A_H()
{
    LD_M_M(_registers.A, _registers.H);
}

void CPU::LD_A_L()
{
    LD_M_M(_registers.A, _registers.L);
}

void CPU::LD_A_MM()
{
}

void CPU::LD_A_A()
{
    LD_M_M(_registers.A, _registers.A);
}

void CPU::ADD_A_B()
{
    ADD_A_M(_registers.B);
}

void CPU::ADD_A_C()
{
    ADD_A_M(_registers.C);
}

void CPU::ADD_A_D()
{
    ADD_A_M(_registers.D);
}

void CPU::ADD_A_E()
{
    ADD_A_M(_registers.E);
}

void CPU::ADD_A_H()
{
    ADD_A_M(_registers.H);
}

void CPU::ADD_A_L()
{
    ADD_A_M(_registers.L);
}

void CPU::ADD_A_MM()
{
}

void CPU::ADD_A_A()
{
    ADD_A_M(_registers.A);
}

void CPU::ADD_A_M(uint8 value)
{
}

void CPU::ADC_A_B()
{
    ADC_A_M(_registers.B);
}

void CPU::ADC_A_C()
```

```
{
    ADC_A_M(_registers.C);
}

void CPU::ADC_A_D()
{
    ADC_A_M(_registers.D);
}

void CPU::ADC_A_E()
{
    ADC_A_M(_registers.E);
}

void CPU::ADC_A_H()
{
    ADC_A_M(_registers.H);
}

void CPU::ADC_A_L()
{
    ADC_A_M(_registers.L);
}

void CPU::ADC_A_MM()
{
}

void CPU::ADC_A_A()
{
    ADC_A_M(_registers.A);
}

void CPU::ADC_A_M(uint8 value)
{
}

void CPU::SUB_A_B()
{
    SUB_A_M(_registers.B);
}

void CPU::SUB_A_C()
{
    SUB_A_M(_registers.C);
}

void CPU::SUB_A_D()
{
    SUB_A_M(_registers.D);
}

void CPU::SUB_A_E()
{
    SUB_A_M(_registers.E);
}
```

```

void CPU::SUB_A_H()
{
    SUB_A_M(_registers.H);
}

void CPU::SUB_A_L()
{
    SUB_A_M(_registers.L);
}

void CPU::SUB_A_MM()
{
}

void CPU::SUB_A_A()
{
    SUB_A_M(_registers.A);
}

void CPU::SUB_A_M(uint8 value)
{
}

void CPU::SBC_A_B()
{
    SBC_A_M(_registers.B);
}

void CPU::SBC_A_C()
{
    SBC_A_M(_registers.C);
}

void CPU::SBC_A_D()
{
    SBC_A_M(_registers.D);
}

void CPU::SBC_A_E()
{
    SBC_A_M(_registers.E);
}

void CPU::SBC_A_H()
{
    SBC_A_M(_registers.H);
}

void CPU::SBC_A_L()
{
    SBC_A_M(_registers.L);
}

void CPU::SBC_A_MM()
{
}

void CPU::SBC_A_A()
{
    SBC_A_M(_registers.A);
}

void CPU::SBC_A_M(uint8 value)
{
}

void CPU::AND_B()
{
    AND_M(_registers.B);
}

void CPU::AND_C()
{
    AND_M(_registers.C);
}

void CPU::AND_D()
{
    AND_M(_registers.D);
}

void CPU::AND_E()
{
    AND_M(_registers.E);
}

void CPU::AND_H()
{
    AND_M(_registers.H);
}

void CPU::AND_L()
{
    AND_M(_registers.L);
}

void CPU::AND_MM()
{
}

void CPU::AND_A()
{
    AND_M(_registers.A);
}

void CPU::AND_M(uint8 value)
{
}

void CPU::XOR_B()
{
    XOR_M(_registers.B);
}

void CPU::XOR_C()
{
    XOR_M(_registers.C);
}

void CPU::XOR_D()
{
    XOR_M(_registers.D);
}

void CPU::XOR_E()
{
    XOR_M(_registers.E);
}

void CPU::XOR_H()
{
    XOR_M(_registers.H);
}

void CPU::XOR_L()
{
    XOR_M(_registers.L);
}

void CPU::XOR_MM()
{
}

void CPU::XOR_A()
{
    XOR_M(_registers.A);
}

void CPU::XOR_M(uint8 value)
{
}

void CPU::OR_B()
{
    OR_M(_registers.B);
}

void CPU::OR_C()
{
    OR_M(_registers.C);
}

void CPU::OR_D()
{
    OR_M(_registers.D);
}

```

```
void CPU::OR_E()
{
    OR_M(_registers.E);
}

void CPU::OR_H()
{
    OR_M(_registers.H);
}

void CPU::OR_L()
{
    OR_M(_registers.L);
}

void CPU::OR_MM()
{
}

void CPU::OR_A()
{
    OR_M(_registers.A);
}

void CPU::OR_M(uint8 value)
{
}

void CPU::CP_B()
{
    CP_M(_registers.B);
}

void CPU::CP_C()
{
    CP_M(_registers.C);
}

void CPU::CP_D()
{
    CP_M(_registers.D);
}

void CPU::CP_E()
{
    CP_M(_registers.E);
}

void CPU::CP_H()
{
    CP_M(_registers.H);
}

void CPU::CP_L()
{
    CP_M(_registers.L);
```

```
}

void CPU::CP_MM()
{
}

void CPU::CP_A()
{
    CP_M(_registers.A);
}

void CPU::CP_M(uint8 value)
{
}

void CPU::RET_NZ()
{
}

void CPU::POP_BC()
{
}

void CPU::JP_NZ_RR()
{
}

void CPU::JP_RR()
{
    uint16 jmpTo = Read16(_registers.PC);
    _registers.PC = jmpTo;
    _hasSetPC = TRUE;
}

void CPU::CALL_NZ_RR()
{
}

void CPU::PUSH_BC()
{
}

void CPU::ADD_A_R()
{
}

void CPU::RST_00H()
{
}

void CPU::RET_Z()
{
}

void CPU::RET()
{
}

void CPU::JP_Z_RR()
{
}

void CPU::PREFIX_CB()
{
}

void CPU::CALL_Z_RR()
{
}

void CPU::CALL_RR()
{
}

void CPU::ADC_A_R()
{
}

void CPU::RST_08H()
{
}

void CPU::RET_NC()
{
}

void CPU::POP_DE()
{
}

void CPU::JP_NC_RR()
{
}

void CPU::CALL_NC_RR()
{
}

void CPU::PUSH_DE()
{
}

void CPU::SUB_R()
{
}

void CPU::RST_10H()
{
}

void CPU::RET_C()
{
}

void CPU::RETI()
{
}

void CPU::JP_C_RR()
{
}

void CPU::CALL_C_RR()
{
}

void CPU::SBC_A_R()
{
}

void CPU::RST_18H()
{
}

void CPU::LDH_R_A()
{
}

void CPU::POP_HL()
{
}

void CPU::LD_C_A2()
{
}

void CPU::PUSH_HL()
{
}

void CPU::AND_R()
{
}

void CPU::RST_20H()
{
}

void CPU::ADD_SP_R()
{
}

void CPU::JP_MM()
{
}

void CPU::LD_RR_A()
{
}

void CPU::XOR_R()
{
}

void CPU::RST_28H()
{
}

void CPU::LDH_A_R()
{
    _registers.A = Read8(0xFF00 + Read8(_registers.PC));
    _registers.tClock.Cycle(3, 12);
}

void CPU::POP_AF()
{
}

void CPU::LD_A_C2()
{
    _registers.A = Read8(0xFF00 + _registers.C);
    _registers.tClock.Cycle(2, 8);
}

void CPU::DI()
{
}

void CPU::PUSH_AF()
{
}

void CPU::OR_R()
{
}

void CPU::RST_30H()
{
}

void CPU::LD_HL_SPandR()
{
}

void CPU::LD_SP_HL()
{
}

void CPU::LD_A_RR()
{
}

void CPU::EI()
{
}

void CPU::CP_R()
{
}

void CPU::RST_38H()
{
}

void CPU::LD_M_M(uint8 & address, uint8 value)
{
    address = value;
    _registers.tClock.Cycle(1, 4);
}

void CPU::INC_M(uint8 & address)
{
    address += 0x01;
    _registers.tClock.Cycle(1, 4);
}

void CPU::DEC_M(uint8 & address)
{
    address -= 0x01;
    _registers.tClock.Cycle(1, 4);
}

void CPU::CB_RLC_B()
{
    CB_RLC8(_registers.B);
}

void CPU::CB_RLC_C()
{
    CB_RLC8(_registers.C);
}

void CPU::CB_RLC_D()
{
    CB_RLC8(_registers.D);
}

void CPU::CB_RLC_E()
{
    CB_RLC8(_registers.E);
}

void CPU::CB_RLC_H()
{
    CB_RLC8(_registers.H);
}

void CPU::CB_RLC_L()
{
    CB_RLC8(_registers.L);
```

```
}

void CPU::CB_RLC_MM()
{
    _registers.tClock.Cycle(2, 16);
}

void CPU::CB_RLC_A()
{
    CB_RLC8(_registers.A);
}

void CPU::CB_RRC_B()
{
    CB_RRC8(_registers.B);
}

void CPU::CB_RRC_C()
{
    CB_RRC8(_registers.C);
}

void CPU::CB_RRC_D()
{
    CB_RRC8(_registers.D);
}

void CPU::CB_RRC_E()
{
    CB_RRC8(_registers.E);
}

void CPU::CB_RRC_H()
{
    CB_RRC8(_registers.H);
}

void CPU::CB_RRC_L()
{
    CB_RRC8(_registers.L);
}

void CPU::CB_RRC_MM()
{
    _registers.tClock.Cycle(2, 16);
}

void CPU::CB_RRC_A()
{
    CB_RRC8(_registers.A);
}

void CPU::CB_RL_B()
{
    CB_RL8(_registers.B);
}
```

```

void CPU::CB_RL_C()
{
    CB_RL8(_registers.C);
}

void CPU::CB_RL_D()
{
    CB_RL8(_registers.D);
}

void CPU::CB_RL_E()
{
    CB_RL8(_registers.E);
}

void CPU::CB_RL_H()
{
    CB_RL8(_registers.H);
}

void CPU::CB_RL_L()
{
    CB_RL8(_registers.L);
}

void CPU::CB_RL_MM()
{
    _registers.tClock.Cycle(2, 16);
}

void CPU::CB_RL_A()
{
    CB_RL8(_registers.A);
}

void CPU::CB_RR_B()
{
    CB_RR8(_registers.B);
}

void CPU::CB_RR_C()
{
    CB_RR8(_registers.C);
}

void CPU::CB_RR_D()
{
    CB_RR8(_registers.D);
}

void CPU::CB_RR_E()
{
    CB_RR8(_registers.E);
}

```

```cpp
void CPU::CB_RR_H()
{
    CB_RR8(_registers.H);
}

void CPU::CB_RR_L()
{
    CB_RR8(_registers.L);
}

void CPU::CB_RR_MM()
{
    _registers.tClock.Cycle(2, 16);
}

void CPU::CB_RR_A()
{
    CB_RR8(_registers.A);
}

void CPU::CB_SLA_B()
{
    CB_SLA8(_registers.B);
}

void CPU::CB_SLA_C()
{
    CB_SLA8(_registers.C);
}

void CPU::CB_SLA_D()
{
    CB_SLA8(_registers.D);
}

void CPU::CB_SLA_E()
{
}

void CPU::CB_SLA_H()
{
    CB_SLA8(_registers.H);
}

void CPU::CB_SLA_L()
{
    CB_SLA8(_registers.L);
}

void CPU::CB_SLA_MM()
{
    _registers.tClock.Cycle(2, 16);
}

void CPU::CB_SLA_A()
{
    CB_SLA8(_registers.A);
}

void CPU::CB_SRA_B()
{
    CB_SRA8(_registers.B);
}

void CPU::CB_SRA_C()
{
    CB_SRA8(_registers.C);
}

void CPU::CB_SRA_D()
{
    CB_SRA8(_registers.D);
}

void CPU::CB_SRA_E()
{
    CB_SRA8(_registers.E);
}

void CPU::CB_SRA_H()
{
    CB_SRA8(_registers.H);
}

void CPU::CB_SRA_L()
{
    CB_SRA8(_registers.L);
}

void CPU::CB_SRA_MM()
{
    _registers.tClock.Cycle(2, 16);
}

void CPU::CB_SRA_A()
{
    CB_SRA8(_registers.A);
}

void CPU::CB_SWAP_B()
{
    CB_SWAP8(_registers.B);
}

void CPU::CB_SWAP_C()
{
    CB_SWAP8(_registers.C);
}

void CPU::CB_SWAP_D()
{
    CB_SWAP8(_registers.D);
```

```
}

void CPU::CB_SWAP_E()
{
    CB_SWAP8(_registers.E);
}

void CPU::CB_SWAP_H()
{
    CB_SWAP8(_registers.H);
}

void CPU::CB_SWAP_L()
{
    CB_SWAP8(_registers.L);
}

void CPU::CB_SWAP_MM()
{
    _registers.tClock.Cycle(2, 16);
}

void CPU::CB_SWAP_A()
{
    CB_SWAP8(_registers.A);
}

void CPU::CB_SRL_B()
{
    CB_SRL8(_registers.B);
}

void CPU::CB_SRL_C()
{
    CB_SRL8(_registers.C);
}

void CPU::CB_SRL_D()
{
    CB_SRL8(_registers.D);
}

void CPU::CB_SRL_E()
{
    CB_SRL8(_registers.E);
}

void CPU::CB_SRL_H()
{
    CB_SRL8(_registers.H);
}

void CPU::CB_SRL_L()
{
    CB_SRL8(_registers.L);
}
```

```cpp

void CPU::CB_SRL_MM()
{
    _registers.tClock.Cycle(2, 16);
}

void CPU::CB_SRL_A()
{
    CB_SRL8(_registers.A);
}

void CPU::CB_RLC8(uint8 & registerRef)
{
    _registers.tClock.Cycle(2, 8);
}

void CPU::CB_RRC8(uint8 & registerRef)
{
    _registers.tClock.Cycle(2, 8);
}

void CPU::CB_RL8(uint8 & registerRef)
{
    _registers.tClock.Cycle(2, 8);
}

void CPU::CB_RR8(uint8 & registerRef)
{
    _registers.tClock.Cycle(2, 8);
}

void CPU::CB_SLA8(uint8 & registerRef)
{
    _registers.tClock.Cycle(2, 8);
}

void CPU::CB_SRA8(uint8 & registerRef)
{
    _registers.tClock.Cycle(2, 8);
}

void CPU::CB_SWAP8(uint8 & registerRef)
{
    _registers.tClock.Cycle(2, 8);
}

void CPU::CB_SRL8(uint8 & registerRef)
{
    _registers.tClock.Cycle(2, 8);
}

void CPU::CB_BIT8(uint8 bit, uint8 & registerRef)
{
    _registers.tClock.Cycle(2, 8);
}

```

```
void CPU::CB_BIT16(uint8 bit, uint16 & registerRef)
{
    _registers.tClock.Cycle(2, 16);
}

void CPU::CB_RES8(uint8 bit, uint8 & registerRef)
{
    _registers.tClock.Cycle(2, 8);
}

void CPU::CB_RES16(uint8 bit, uint16 & registerRef)
{
    _registers.tClock.Cycle(2, 16);
}

void CPU::CB_SET8(uint8 bit, uint8 & registerRef)
{
    _registers.tClock.Cycle(2, 8);
}

void CPU::CB_SET16(uint8 bit, uint16 & registerRef)
{
    _registers.tClock.Cycle(2, 16);
}

```