

```

1  #include "CPU.h"
2
3  Input* input;
4
5  CPU::CPU(MMU* mmu)
6  {
7      _mmu = mmu;
8      input = new Input();
9      Reset();
10
11 }
12
13 CPU::~CPU()
14 {
15 }
16
17 void CPU::Reset()
18 {
19     _clock.Cycle(0, 0);
20     _registers.tClock.Cycle(0, 0);
21
22     _registers.A = 0;
23     _registers.F = 0;
24     _registers.B = 0;
25     _registers.C = 0;
26     _registers.D = 0;
27     _registers.E = 0;
28     _registers.H = 0;
29     _registers.L = 0;
30     _registers.SP = 0xCFFF;
31     _registers.PC = 0x0100;
32
33     for (uint8 i = 0; i < 0xFF; i++)
34     {
35         _opcodeTable[i] = &CPU::NOP;
36     }
37
38     _opcodeTable[0x01] = &CPU::LD_BC_RR;           //0x01 LD BC,d16
39     _opcodeTable[0x02] = &CPU::LD_BC_A;           //0x02 LD (BC),A
40     _opcodeTable[0x03] = &CPU::INC_BC;           //0x03 INC BC
41     _opcodeTable[0x04] = &CPU::INC_B;           //0x04 INC B
42     _opcodeTable[0x05] = &CPU::DEC_B;           //0x05 DEC B
43     _opcodeTable[0x06] = &CPU::LD_B_R;           //0x06 LD B,d8
44     _opcodeTable[0x07] = &CPU::RL_CA;           //0x07
45     _opcodeTable[0x08] = &CPU::LD_RR_SP;         //0x08
46     _opcodeTable[0x09] = &CPU::ADD_HL_BC;        //0x09
47     _opcodeTable[0x0A] = &CPU::LD_A_BC;         //0x0A
48     _opcodeTable[0x0B] = &CPU::DEC_BC;          //0x0B
49     _opcodeTable[0x0C] = &CPU::INC_C;           //0x0C
50     _opcodeTable[0x0D] = &CPU::DEC_C;           //0x0D
51     _opcodeTable[0x0E] = &CPU::LD_C_R;         //0x0E
52     _opcodeTable[0x0F] = &CPU::RR_CA;           //0x0F
53
54     _opcodeTable[0x10] = &CPU::STOP;            //0x10
55     _opcodeTable[0x11] = &CPU::LD_DE_RR;        //0x11
56     _opcodeTable[0x12] = &CPU::LD_DE_A;         //0x12

```

```
57     _opcodeTable[0x13] = &CPU::INC_DE;           //0x13
58     _opcodeTable[0x14] = &CPU::INC_D;            //0x14
59     _opcodeTable[0x15] = &CPU::DEC_D;            //0x15
60     _opcodeTable[0x16] = &CPU::LD_D_R;           //0x16
61     _opcodeTable[0x17] = &CPU::RL_A;             //0x17
62     _opcodeTable[0x18] = &CPU::JR_R;             //0x18
63     _opcodeTable[0x19] = &CPU::ADD_HL_DE;         //0x19
64     _opcodeTable[0x1A] = &CPU::LD_A_DE;          //0x1A
65     _opcodeTable[0x1B] = &CPU::DEC_DE;           //0x1B
66     _opcodeTable[0x1C] = &CPU::INC_E;            //0x1C
67     _opcodeTable[0x1D] = &CPU::DEC_E;            //0x1D
68     _opcodeTable[0x1E] = &CPU::LD_E_R;           //0x1E
69     _opcodeTable[0x1F] = &CPU::RR_A;             //0x1F
70
71     _opcodeTable[0x20] = &CPU::JR_NZ_R;          //0x20
72     _opcodeTable[0x21] = &CPU::LD_HL_RR;         //0x21
73     _opcodeTable[0x22] = &CPU::LD_MM_PLUS_A;     //0x22
74     _opcodeTable[0x23] = &CPU::INC_HL;           //0x23
75     _opcodeTable[0x24] = &CPU::INC_H;            //0x24
76     _opcodeTable[0x25] = &CPU::DEC_H;            //0x25
77     _opcodeTable[0x26] = &CPU::LD_H_R;           //0x26
78     _opcodeTable[0x27] = &CPU::DA_A;             //0x27
79     _opcodeTable[0x28] = &CPU::JR_Z_R;           //0x28
80     _opcodeTable[0x29] = &CPU::ADD_HL_HL;        //0x29
81     _opcodeTable[0x2A] = &CPU::LD_A_MM_PLUS;     //0x2A
82     _opcodeTable[0x2B] = &CPU::DEC_HL;           //0x2B
83     _opcodeTable[0x2C] = &CPU::INC_L;            //0x2C
84     _opcodeTable[0x2D] = &CPU::DEC_L;            //0x2D
85     _opcodeTable[0x2E] = &CPU::LD_L_R;           //0x2E
86     _opcodeTable[0x2F] = &CPU::CPL;              //0x2F
87
88     _opcodeTable[0x30] = &CPU::JR_NC_R;          //0x30
89     _opcodeTable[0x31] = &CPU::LD_SP_RR;         //0x31
90     _opcodeTable[0x32] = &CPU::LD_MM_MIN_A;      //0x32
91     _opcodeTable[0x33] = &CPU::INC_SP;           //0x33
92     _opcodeTable[0x34] = &CPU::INC_MM;           //0x34
93     _opcodeTable[0x35] = &CPU::DEC_MM;           //0x35
94     _opcodeTable[0x36] = &CPU::LD_MM_R;          //0x36
95     _opcodeTable[0x37] = &CPU::S_CF;             //0x37
96     _opcodeTable[0x38] = &CPU::JR_C_R;           //0x38
97     _opcodeTable[0x39] = &CPU::ADD_HL_SP;        //0x39
98     _opcodeTable[0x3A] = &CPU::LD_A_MM_MIN;      //0x3A
99     _opcodeTable[0x3B] = &CPU::DEC_SP;           //0x3B
100    _opcodeTable[0x3C] = &CPU::INC_A;             //0x3C
101    _opcodeTable[0x3D] = &CPU::DEC_A;             //0x3D
102    _opcodeTable[0x3E] = &CPU::LD_A_R;            //0x3E
103    _opcodeTable[0x3F] = &CPU::C_CF;              //0x3F
104
105    _opcodeTable[0x40] = &CPU::LD_B_B;            //0x40
106    _opcodeTable[0x41] = &CPU::LD_B_C;            //0x41
107    _opcodeTable[0x42] = &CPU::LD_B_D;            //0x42
108    _opcodeTable[0x43] = &CPU::LD_B_E;            //0x43
109    _opcodeTable[0x44] = &CPU::LD_B_H;            //0x44
110    _opcodeTable[0x45] = &CPU::LD_B_L;            //0x45
111    _opcodeTable[0x46] = &CPU::LD_B_MM;           //0x46
112    _opcodeTable[0x47] = &CPU::LD_B_A;            //0x47
```

```
113     _opcodeTable[0x48] = &CPU::LD_C_B;           //0x48
114     _opcodeTable[0x49] = &CPU::LD_C_C;           //0x49
115     _opcodeTable[0x4A] = &CPU::LD_C_D;           //0x4A
116     _opcodeTable[0x4B] = &CPU::LD_C_E;           //0x4B
117     _opcodeTable[0x4C] = &CPU::LD_C_H;           //0x4C
118     _opcodeTable[0x4D] = &CPU::LD_C_L;           //0x4D
119     _opcodeTable[0x4E] = &CPU::LD_C_MM;          //0x4E
120     _opcodeTable[0x4F] = &CPU::LD_C_A;           //0x4F
121
122     _opcodeTable[0x50] = &CPU::LD_D_B;           //0x50
123     _opcodeTable[0x51] = &CPU::LD_D_C;           //0x51
124     _opcodeTable[0x52] = &CPU::LD_D_D;           //0x52
125     _opcodeTable[0x53] = &CPU::LD_D_E;           //0x53
126     _opcodeTable[0x54] = &CPU::LD_D_H;           //0x54
127     _opcodeTable[0x55] = &CPU::LD_D_L;           //0x55
128     _opcodeTable[0x56] = &CPU::LD_D_MM;          //0x56
129     _opcodeTable[0x57] = &CPU::LD_D_A;           //0x57
130     _opcodeTable[0x58] = &CPU::LD_E_B;           //0x58
131     _opcodeTable[0x59] = &CPU::LD_E_C;           //0x59
132     _opcodeTable[0x5A] = &CPU::LD_E_D;           //0x5A
133     _opcodeTable[0x5B] = &CPU::LD_E_E;           //0x5B
134     _opcodeTable[0x5C] = &CPU::LD_E_H;           //0x5C
135     _opcodeTable[0x5D] = &CPU::LD_E_L;           //0x5D
136     _opcodeTable[0x5E] = &CPU::LD_E_MM;          //0x5E
137     _opcodeTable[0x5F] = &CPU::LD_E_A;           //0x5F
138
139     _opcodeTable[0x60] = &CPU::LD_H_B;           //0x60
140     _opcodeTable[0x61] = &CPU::LD_H_C;           //0x61
141     _opcodeTable[0x62] = &CPU::LD_H_D;           //0x62
142     _opcodeTable[0x63] = &CPU::LD_H_E;           //0x63
143     _opcodeTable[0x64] = &CPU::LD_H_H;           //0x64
144     _opcodeTable[0x65] = &CPU::LD_H_L;           //0x65
145     _opcodeTable[0x66] = &CPU::LD_H_MM;          //0x66
146     _opcodeTable[0x67] = &CPU::LD_H_A;           //0x67
147     _opcodeTable[0x68] = &CPU::LD_L_B;           //0x68
148     _opcodeTable[0x69] = &CPU::LD_L_C;           //0x69
149     _opcodeTable[0x6A] = &CPU::LD_L_D;           //0x6A
150     _opcodeTable[0x6B] = &CPU::LD_L_E;           //0x6B
151     _opcodeTable[0x6C] = &CPU::LD_L_H;           //0x6C
152     _opcodeTable[0x6D] = &CPU::LD_L_L;           //0x6D
153     _opcodeTable[0x6E] = &CPU::LD_L_MM;          //0x6E
154     _opcodeTable[0x6F] = &CPU::LD_L_A;           //0x6F
155
156     _opcodeTable[0x70] = &CPU::LD_MM_B;          //0x70
157     _opcodeTable[0x71] = &CPU::LD_MM_C;          //0x71
158     _opcodeTable[0x72] = &CPU::LD_MM_D;          //0x72
159     _opcodeTable[0x73] = &CPU::LD_MM_E;          //0x73
160     _opcodeTable[0x74] = &CPU::LD_MM_H;          //0x74
161     _opcodeTable[0x75] = &CPU::LD_MM_L;          //0x75
162     _opcodeTable[0x76] = &CPU::HALT;              //0x76
163     _opcodeTable[0x77] = &CPU::LD_MM_A;          //0x77
164     _opcodeTable[0x78] = &CPU::LD_A_B;           //0x78
165     _opcodeTable[0x79] = &CPU::LD_A_C;           //0x79
166     _opcodeTable[0x7A] = &CPU::LD_A_D;           //0x7A
167     _opcodeTable[0x7B] = &CPU::LD_A_E;           //0x7B
168     _opcodeTable[0x7C] = &CPU::LD_A_H;           //0x7C
```

```

169     _opcodeTable[0x7D] = &CPU::LD_A_L;           //0x7D
170     _opcodeTable[0x7E] = &CPU::LD_A_MM;          //0x7E
171     _opcodeTable[0x7F] = &CPU::LD_A_A;           //0x7F
172
173
174     _opcodeTable[0x80] = &CPU::ADD_A_B;           //0x80
175     _opcodeTable[0x81] = &CPU::ADD_A_C;           //0x81
176     _opcodeTable[0x82] = &CPU::ADD_A_D;           //0x82
177     _opcodeTable[0x83] = &CPU::ADD_A_E;           //0x83
178     _opcodeTable[0x84] = &CPU::ADD_A_H;           //0x84
179     _opcodeTable[0x85] = &CPU::ADD_A_L;           //0x85
180     _opcodeTable[0x86] = &CPU::ADD_A_MM;          //0x86
181     _opcodeTable[0x87] = &CPU::ADD_A_A;           //0x87
182     _opcodeTable[0x88] = &CPU::ADC_A_B;           //0x88
183     _opcodeTable[0x89] = &CPU::ADC_A_C;           //0x89
184     _opcodeTable[0x8A] = &CPU::ADC_A_D;           //0x8A
185     _opcodeTable[0x8B] = &CPU::ADC_A_E;           //0x8B
186     _opcodeTable[0x8C] = &CPU::ADC_A_H;           //0x8C
187     _opcodeTable[0x8D] = &CPU::ADC_A_L;           //0x8D
188     _opcodeTable[0x8E] = &CPU::ADC_A_MM;          //0x8E
189     _opcodeTable[0x8F] = &CPU::ADC_A_A;           //0x8F
190
191     _opcodeTable[0x90] = &CPU::SUB_A_B;           //0x90
192     _opcodeTable[0x91] = &CPU::SUB_A_C;           //0x91
193     _opcodeTable[0x92] = &CPU::SUB_A_D;           //0x92
194     _opcodeTable[0x93] = &CPU::SUB_A_E;           //0x93
195     _opcodeTable[0x94] = &CPU::SUB_A_H;           //0x94
196     _opcodeTable[0x95] = &CPU::SUB_A_L;           //0x95
197     _opcodeTable[0x96] = &CPU::SUB_A_MM;          //0x96
198     _opcodeTable[0x97] = &CPU::SUB_A_A;           //0x97
199     _opcodeTable[0x98] = &CPU::SBC_A_B;           //0x98
200     _opcodeTable[0x99] = &CPU::SBC_A_C;           //0x99
201     _opcodeTable[0x9A] = &CPU::SBC_A_D;           //0x9A
202     _opcodeTable[0x9B] = &CPU::SBC_A_E;           //0x9B
203     _opcodeTable[0x9C] = &CPU::SBC_A_H;           //0x9C
204     _opcodeTable[0x9D] = &CPU::SBC_A_L;           //0x9D
205     _opcodeTable[0x9E] = &CPU::SBC_A_MM;          //0x9E
206     _opcodeTable[0x9F] = &CPU::SBC_A_A;           //0x9F
207
208     _opcodeTable[0xA0] = &CPU::AND_B;             //0xA0
209     _opcodeTable[0xA1] = &CPU::AND_C;             //0xA1
210     _opcodeTable[0xA2] = &CPU::AND_D;             //0xA2
211     _opcodeTable[0xA3] = &CPU::AND_E;             //0xA3
212     _opcodeTable[0xA4] = &CPU::AND_H;             //0xA4
213     _opcodeTable[0xA5] = &CPU::AND_L;             //0xA5
214     _opcodeTable[0xA6] = &CPU::AND_MM;            //0xA6
215     _opcodeTable[0xA7] = &CPU::AND_A;             //0xA7
216     _opcodeTable[0xA8] = &CPU::XOR_B;             //0xA8
217     _opcodeTable[0xA9] = &CPU::XOR_C;             //0xA9
218     _opcodeTable[0xAA] = &CPU::XOR_D;             //0xAA
219     _opcodeTable[0xAB] = &CPU::XOR_E;             //0xAB
220     _opcodeTable[0xAC] = &CPU::XOR_H;             //0xAC
221     _opcodeTable[0xAD] = &CPU::XOR_L;             //0xAD
222     _opcodeTable[0xAE] = &CPU::XOR_MM;            //0xAE
223     _opcodeTable[0xAF] = &CPU::XOR_A;             //0xAF
224

```

```
225 _opcodeTable[0xB0] = &CPU::OR_B;           //0xB0
226 _opcodeTable[0xB1] = &CPU::OR_C;           //0xB1
227 _opcodeTable[0xB2] = &CPU::OR_D;           //0xB2
228 _opcodeTable[0xB3] = &CPU::OR_E;           //0xB3
229 _opcodeTable[0xB4] = &CPU::OR_H;           //0xB4
230 _opcodeTable[0xB5] = &CPU::OR_L;           //0xB5
231 _opcodeTable[0xB6] = &CPU::OR_MM;          //0xB6
232 _opcodeTable[0xB7] = &CPU::OR_A;           //0xB7
233 _opcodeTable[0xB8] = &CPU::CP_B;           //0xB8
234 _opcodeTable[0xB9] = &CPU::CP_C;           //0xB9
235 _opcodeTable[0xBA] = &CPU::CP_D;           //0xBA
236 _opcodeTable[0xBB] = &CPU::CP_E;           //0xBB
237 _opcodeTable[0xBC] = &CPU::CP_H;           //0xBC
238 _opcodeTable[0xBD] = &CPU::CP_L;           //0xBD
239 _opcodeTable[0xBE] = &CPU::CP_MM;          //0xBE
240 _opcodeTable[0xBF] = &CPU::CP_A;           //0xBF
241
242 _opcodeTable[0xC0] = &CPU::RET_NZ;          //0xC0
243 _opcodeTable[0xC1] = &CPU::POP_BC;          //0xC1
244 _opcodeTable[0xC2] = &CPU::JP_NZ_RR;        //0xC2
245 _opcodeTable[0xC3] = &CPU::JP_RR;          //0xC3
246 _opcodeTable[0xC4] = &CPU::CALL_NZ_RR;      //0xC4
247 _opcodeTable[0xC5] = &CPU::PUSH_BC;         //0xC5
248 _opcodeTable[0xC6] = &CPU::ADD_A_R;         //0xC6
249 _opcodeTable[0xC7] = &CPU::RST_00H;         //0xC7
250 _opcodeTable[0xC8] = &CPU::RET_Z;           //0xC8
251 _opcodeTable[0xC9] = &CPU::RET;            //0xC9
252 _opcodeTable[0xCA] = &CPU::JP_Z_RR;         //0xCA
253 _opcodeTable[0xCB] = &CPU::PREFIX_CB;       //0xCB
254 _opcodeTable[0xCC] = &CPU::CALL_Z_RR;       //0xCC
255 _opcodeTable[0xCD] = &CPU::CALL_RR;         //0xCD
256 _opcodeTable[0xCE] = &CPU::ADC_A_R;         //0xCE
257 _opcodeTable[0xCF] = &CPU::RST_08H;         //0xCF
258
259 _opcodeTable[0xD0] = &CPU::RET_NC;          //0xD0
260 _opcodeTable[0xD1] = &CPU::POP_DE;          //0xD1
261 _opcodeTable[0xD2] = &CPU::JP_NC_RR;        //0xD2
262 _opcodeTable[0xD4] = &CPU::CALL_NC_RR;      //0xD4
263 _opcodeTable[0xD5] = &CPU::PUSH_DE;         //0xD5
264 _opcodeTable[0xD6] = &CPU::SUB_R;           //0xD6
265 _opcodeTable[0xD7] = &CPU::RST_10H;         //0xD7
266 _opcodeTable[0xD8] = &CPU::RET_C;           //0xD8
267 _opcodeTable[0xD9] = &CPU::RETI;            //0xD9
268 _opcodeTable[0xDA] = &CPU::JP_C_RR;         //0xDA
269 _opcodeTable[0xDC] = &CPU::CALL_C_RR;       //0xDC
270 _opcodeTable[0xDE] = &CPU::SBC_A_R;         //0xDE
271 _opcodeTable[0xDF] = &CPU::RST_18H;         //0xDF
272
273 _opcodeTable[0xE0] = &CPU::LDH_R_A;         //0xE0
274 _opcodeTable[0xE1] = &CPU::POP_HL;          //0xE1
275 _opcodeTable[0xE2] = &CPU::LD_C_A2;        //0xE2
276 _opcodeTable[0xE5] = &CPU::PUSH_HL;         //0xE5
277 _opcodeTable[0xE6] = &CPU::AND_R;           //0xE6
278 _opcodeTable[0xE7] = &CPU::RST_20H;         //0xE7
279 _opcodeTable[0xE8] = &CPU::ADD_SP_R;        //0xE8
280 _opcodeTable[0xE9] = &CPU::JP_MM;          //0xE9
```

```

281     _opcodeTable[0xEA] = &CPU::LD_RR_A;      //0xEA
282     _opcodeTable[0xEE] = &CPU::XOR_R;        //0xEE
283     _opcodeTable[0xEF] = &CPU::RST_28H;      //0xEF
284
285     _opcodeTable[0xF0] = &CPU::LDH_A_R;      //0xF0
286     _opcodeTable[0xF1] = &CPU::POP_AF;       //0xF1
287     _opcodeTable[0xF2] = &CPU::LD_A_C2;     //0xF2
288     _opcodeTable[0xF3] = &CPU::DI;          //0xF3
289     _opcodeTable[0xF5] = &CPU::PUSH_AF;      //0xF5
290     _opcodeTable[0xF6] = &CPU::OR_R;         //0xF6
291     _opcodeTable[0xF7] = &CPU::RST_30H;      //0xF7
292     _opcodeTable[0xF8] = &CPU::LD_HL_SPandR; //0xF8
293     _opcodeTable[0xF9] = &CPU::LD_SP_HL;     //0xF9
294     _opcodeTable[0xFA] = &CPU::LD_A_RR;      //0xFA
295     _opcodeTable[0xFB] = &CPU::EI;          //0xFB
296     _opcodeTable[0xFE] = &CPU::CP_R;         //0xFE
297     _opcodeTable[0xFF] = &CPU::RST_38H;      //0xFF
298 }
299
300 bool CPU::Run()
301 {
302     //(expected sequence)
303     BOOL runningState = TRUE;
304
305     /*
306     NOP, JP_MM($0150), JP_MM($0185), LD_A_R(A=($180)=0x03),
307     DI, LDH_R_A($FF0F <- A), LDH_R_A($FFF <- A), LD_A_R(a <- 0x40),
308     LDH_R_A($FF41 <- 0x40), XOR_A(A <- 0), LDH_R_A($FF42 <- 0x00),
309     LDH_R_A($FF43 <- 0x00),LDH_R_A($FF44 <- 0x00),
310     */
311     uint8 pc = Read8(_registers.PC);
312     _registers.PC += 0x0001;
313     if (pc == 0xCB)
314     {
315         //cb
316         uint8 pc = Read8(_registers.PC + 1);
317         _cbOpcodeTable[pc];
318     }
319     else
320     {
321         (this->*_opcodeTable[pc])();
322     }
323
324     pc += _registers.tClock.byte_call_cycles;
325     if (_hasSetPC == FALSE)
326     {
327         _registers.PC = pc;
328     }
329
330     //reset
331     _registers.tClock.Cycle(0, 0);
332     _hasSetPC = FALSE;
333
334     if (input->IsExit()) { runningState = FALSE; }
335     return runningState;
336 }

```

```
337
338 uint8 CPU::Read8(uint16 address)
339 {
340     return _mmu->Read8(address);
341 }
342
343 uint16 CPU::Read16(uint16 address)
344 {
345     return _mmu->Read16(address);
346 }
347
348 void CPU::Write8(uint16 address, uint8 value)
349 {
350     return _mmu->Write8(address, value);
351 }
352
353 void CPU::Write16(uint16 address, uint16 value)
354 {
355     return _mmu->Write16(address, value);
356 }
357
358 void CPU::Set_Z(BOOL value)
359 {
360     Set_Bit(_registers.F, 7, value);
361 }
362
363 void CPU::Set_N(BOOL value)
364 {
365     Set_Bit(_registers.F, 6, value);
366 }
367
368 void CPU::Set_H(BOOL value)
369 {
370     Set_Bit(_registers.F, 5, value);
371 }
372 void CPU::Set_C(BOOL value)
373 {
374     Set_Bit(_registers.F, 4, value);
375 }
376
377 void CPU::NOP()
378 {
379     _registers.tClock.Cycle(1, 4);
380 }
381
382 void CPU::LD_BC_RR()
383 {
384     Write8(_registers.BC(), Read8(_registers.PC));
385     _registers.tClock.Cycle(3, 12);
386 }
387
388 void CPU::LD_BC_A()
389 {
390     Write8(_registers.BC(), _registers.A);
391     _registers.tClock.Cycle(1, 8);
392 }
```



```
393
394 void CPU::INC_BC()
395 {
396     uint16 tempBC = _registers.BC();
397     tempBC += 0x0001;
398     _registers.BC(tempBC);
399
400     _registers.tClock.Cycle(1, 8);
401 }
402
403 void CPU::INC_B()
404 {
405     INC_M(_registers.B);
406 }
407
408 void CPU::DEC_B()
409 {
410     DEC_M(_registers.B);
411 }
412
413 void CPU::LD_B_R()
414 {
415     _registers.B = Read8(_registers.PC);
416     _registers.tClock.Cycle(2, 8);
417 }
418
419 void CPU::RL_CA()
420 {
421     uint8 oldA = _registers.A;
422     _registers.A = _registers.A << 1;
423
424     Set_Z(_registers.A == 0);
425     Set_N(FALSE);
426     Set_H(FALSE);
427     Set_C(Get_Bit(oldA, 7));
428
429     _registers.tClock.Cycle(1, 4);
430 }
431
432 void CPU::LD_RR_SP()
433 {
434     Write8(Read16(_registers.PC), Read8(_registers.SP));
435     _registers.tClock.Cycle(3, 20);
436 }
437
438 void CPU::ADD_HL_BC()
439 {
440     _registers.HL(_registers.HL() + _registers.BC());
441     _registers.tClock.Cycle(1, 8);
442 }
443
444 void CPU::LD_A_BC()
445 {
446     _registers.A = Read8(_registers.BC());
447     _registers.tClock.Cycle(1, 8);
448 }
```



```
449
450 void CPU::DEC_BC()
451 {
452     uint16 tempBC = _registers.BC();
453     tempBC -= 0x0001;
454     _registers.BC(tempBC);
455
456     _registers.tClock.Cycle(1, 8);
457 }
458
459 void CPU::INC_C()
460 {
461     INC_M(_registers.C);
462 }
463
464 void CPU::DEC_C()
465 {
466     DEC_M(_registers.C);
467 }
468
469 void CPU::LD_C_R()
470 {
471     _registers.C = Read8(_registers.PC);
472     _registers.tClock.Cycle(2, 8);
473 }
474
475 void CPU::RR_CA()
476 {
477     uint8 oldA = _registers.A;
478     _registers.A = _registers.A >> 1;
479
480     Set_Z(_registers.A == 0);
481     Set_N(FALSE);
482     Set_H(FALSE);
483     Set_C(Get_Bit(oldA, 0));
484
485     _registers.tClock.Cycle(1, 4);
486 }
487
488 void CPU::STOP()
489 {
490     //TODO: STOP
491     _registers.tClock.Cycle(2, 4);
492 }
493
494 void CPU::LD_DE_RR()
495 {
496     uint16 tempDE = _registers.DE();
497     tempDE = Read16(_registers.PC);
498     _registers.DE(tempDE);
499     _registers.tClock.Cycle(3, 12);
500 }
501
502 void CPU::LD_DE_A()
503 {
504     Write8(Read16(_registers.DE()), _registers.A);
```

```
505     _registers.tClock.Cycle(1, 8);
506 }
507
508 void CPU::INC_DE()
509 {
510     uint16 tempDE = _registers.DE();
511     tempDE += 0x0001;
512     _registers.DE(tempDE);
513     _registers.tClock.Cycle(1, 8);
514 }
515
516 void CPU::INC_D()
517 {
518     _registers.D += 0x01;
519     _registers.tClock.Cycle(1, 4);
520 }
521
522 void CPU::DEC_D()
523 {
524     _registers.D -= 0x01;
525     _registers.tClock.Cycle(1, 4);
526 }
527
528 void CPU::LD_D_R()
529 {
530     _registers.D = Read8(_registers.PC);
531     _registers.tClock.Cycle(2, 8);
532 }
533
534 void CPU::RL_A()
535 {
536     uint8 oldA = _registers.A;
537     _registers.A = _registers.A << 1;
538     Set_Bit(_registers.A, 0, Get_Bit(_registers.F, 5));
539     Set_C(Get_Bit(oldA, 0));
540     Set_Z(_registers.A == 0);
541     Set_N(0);
542     Set_H(0);
543     _registers.tClock.Cycle(1, 4);
544 }
545
546 void CPU::JR_R()
547 {
548     _registers.PC += Read8(_registers.PC);
549     _hasSetPC = TRUE;
550     _registers.tClock.Cycle(2, 12);
551 }
552
553 void CPU::ADD_HL_DE()
554 {
555     uint16 tempHL = _registers.HL();
556     tempHL += _registers.DE();
557     _registers.HL(tempHL);
558     Set_N(FALSE);
559     Set_H(Get_Bit(_registers.HL(), 11));
560 }
```

```
561     _registers.tClock.Cycle(1, 8);
562 }
563
564 void CPU::LD_A_DE()
565 {
566     _registers.tClock.Cycle(1, 8);
567 }
568
569 void CPU::DEC_DE()
570 {
571     _registers.tClock.Cycle(1, 8);
572 }
573
574 void CPU::INC_E()
575 {
576     _registers.tClock.Cycle(1, 4);
577 }
578
579 void CPU::DEC_E()
580 {
581     _registers.tClock.Cycle(1, 4);
582 }
583
584 void CPU::LD_E_R()
585 {
586     _registers.tClock.Cycle(2, 8);
587 }
588
589 void CPU::RR_A()
590 {
591     _registers.tClock.Cycle(1, 4);
592 }
593
594 void CPU::JR_NZ_R()
595 {
596     //(depending on result)
597     _registers.tClock.Cycle(2, 12);
598     _registers.tClock.Cycle(2, 8);
599 }
600
601 void CPU::LD_HL_RR()
602 {
603     _registers.tClock.Cycle(1, 4);
604 }
605
606 void CPU::LD_MM_PLUS_A()
607 {
608     _registers.tClock.Cycle(1, 8);
609 }
610
611 void CPU::INC_HL()
612 {
613     _registers.tClock.Cycle(1, 8);
614 }
615
616 void CPU::INC_H()
```

```
617 {
618     _registers.tClock.Cycle(1, 4);
619 }
620
621 void CPU::DEC_H()
622 {
623     _registers.tClock.Cycle(1, 4);
624 }
625
626 void CPU::LD_H_R()
627 {
628     _registers.tClock.Cycle(2, 8);
629 }
630
631 void CPU::DA_A()
632 {
633     _registers.tClock.Cycle(1, 4);
634 }
635
636 void CPU::JR_Z_R()
637 {
638     //(depending on result)
639     _registers.tClock.Cycle(2, 12);
640     _registers.tClock.Cycle(2, 8);
641 }
642
643 void CPU::ADD_HL_HL()
644 {
645     _registers.tClock.Cycle(1, 8);
646 }
647
648 void CPU::LD_A_MM_PLUS()
649 {
650     _registers.tClock.Cycle(1, 8);
651 }
652
653 void CPU::DEC_HL()
654 {
655     _registers.tClock.Cycle(1, 8);
656 }
657
658 void CPU::INC_L()
659 {
660     _registers.tClock.Cycle(1, 4);
661 }
662
663 void CPU::DEC_L()
664 {
665     _registers.tClock.Cycle(1, 4);
666 }
667
668 void CPU::LD_L_R()
669 {
670     _registers.tClock.Cycle(2, 8);
671 }
672
```

```
673 void CPU::CPL()
674 {
675     _registers.tClock.Cycle(1, 4);
676 }
677
678 void CPU::JR_NC_R()
679 {
680     //(depending on result)
681     _registers.tClock.Cycle(2, 12);
682     _registers.tClock.Cycle(2, 8);
683 }
684
685 void CPU::LD_SP_RR()
686 {
687     _registers.tClock.Cycle(3, 12);
688 }
689
690 void CPU::LD_MM_MIN_A()
691 {
692     _registers.tClock.Cycle(1, 8);
693 }
694
695 void CPU::INC_SP()
696 {
697     _registers.tClock.Cycle(1, 8);
698 }
699
700 void CPU::INC_MM()
701 {
702     _registers.tClock.Cycle(1, 12);
703 }
704
705 void CPU::DEC_MM()
706 {
707     _registers.tClock.Cycle(1, 12);
708 }
709
710 void CPU::LD_MM_R()
711 {
712     _registers.tClock.Cycle(2, 12);
713 }
714
715 void CPU::S_CF()
716 {
717     _registers.tClock.Cycle(1, 4);
718 }
719
720 void CPU::JR_C_R()
721 {
722     //(depending on result)
723     _registers.tClock.Cycle(2, 12);
724     _registers.tClock.Cycle(2, 8);
725 }
726
727 void CPU::ADD_HL_SP()
728 {
```

```
729     _registers.tClock.Cycle(1, 8);
730 }
731
732 void CPU::LD_MM_MIN()
733 {
734     _registers.A = Read8(_registers.HL() - 0x0001);
735     _registers.tClock.Cycle(1, 8);
736 }
737
738 void CPU::DEC_SP()
739 {
740     _registers.SP--;
741     _registers.tClock.Cycle(1, 8);
742 }
743
744 void CPU::INC_A()
745 {
746     _registers.A++;
747     _registers.tClock.Cycle(1, 4);
748 }
749
750 void CPU::DEC_A()
751 {
752     _registers.A--;
753     _registers.tClock.Cycle(1, 4);
754 }
755
756 void CPU::LD_AR()
757 {
758     _registers.A = Read8(_registers.PC);
759     _registers.tClock.Cycle(2, 8);
760 }
761
762 void CPU::C_CF()
763 {
764     _registers.tClock.Cycle(1, 4);
765 }
766
767 void CPU::LD_BB()
768 {
769     LD_MM(_registers.B, _registers.B);
770 }
771
772 void CPU::LD_BC()
773 {
774     LD_MM(_registers.B, _registers.C);
775 }
776
777 void CPU::LD_BD()
778 {
779     LD_MM(_registers.B, _registers.D);
780 }
781
782 void CPU::LD_BE()
783 {
784     LD_MM(_registers.B, _registers.E);
```

```
785 }
786
787 void CPU::LD_B_H()
788 {
789     LD_M_M(_registers.B, _registers.H);
790 }
791
792 void CPU::LD_B_L()
793 {
794     LD_M_M(_registers.B, _registers.L);
795 }
796
797 void CPU::LD_B_MM()
798 {
799 }
800
801 void CPU::LD_B_A()
802 {
803     LD_M_M(_registers.B, _registers.A);
804 }
805
806 void CPU::LD_C_B()
807 {
808     LD_M_M(_registers.C, _registers.B);
809 }
810
811 void CPU::LD_C_C()
812 {
813     LD_M_M(_registers.C, _registers.C);
814 }
815
816 void CPU::LD_C_D()
817 {
818     LD_M_M(_registers.C, _registers.D);
819 }
820
821 void CPU::LD_C_E()
822 {
823     LD_M_M(_registers.C, _registers.E);
824 }
825
826 void CPU::LD_C_H()
827 {
828     LD_M_M(_registers.C, _registers.H);
829 }
830
831 void CPU::LD_C_L()
832 {
833     LD_M_M(_registers.C, _registers.L);
834 }
835
836 void CPU::LD_C_MM()
837 {
838 }
839
840 void CPU::LD_C_A()
```



```
841 {
842     LD_M_M(_registers.C, _registers.A);
843 }
844
845 void CPU::LD_D_B()
846 {
847     LD_M_M(_registers.D, _registers.B);
848 }
849
850 void CPU::LD_D_C()
851 {
852     LD_M_M(_registers.D, _registers.C);
853 }
854
855 void CPU::LD_D_D()
856 {
857     LD_M_M(_registers.D, _registers.D);
858 }
859
860 void CPU::LD_D_E()
861 {
862     LD_M_M(_registers.D, _registers.E);
863 }
864
865 void CPU::LD_D_H()
866 {
867     LD_M_M(_registers.D, _registers.H);
868 }
869
870 void CPU::LD_D_L()
871 {
872     LD_M_M(_registers.D, _registers.L);
873 }
874
875 void CPU::LD_D_MM()
876 {
877 }
878
879 void CPU::LD_D_A()
880 {
881     LD_M_M(_registers.D, _registers.A);
882 }
883
884 void CPU::LD_E_B()
885 {
886     LD_M_M(_registers.E, _registers.B);
887 }
888
889 void CPU::LD_E_C()
890 {
891     LD_M_M(_registers.E, _registers.C);
892 }
893
894 void CPU::LD_E_D()
895 {
896     LD_M_M(_registers.E, _registers.D);
```

```
897 }
898
899 void CPU::LD_E_E()
900 {
901     LD_MM(_registers.E, _registers.E);
902 }
903
904 void CPU::LD_E_H()
905 {
906     LD_MM(_registers.E, _registers.H);
907 }
908
909 void CPU::LD_E_L()
910 {
911     LD_MM(_registers.E, _registers.L);
912 }
913
914 void CPU::LD_E_MM()
915 {
916 }
917
918 void CPU::LD_E_A()
919 {
920     LD_MM(_registers.E, _registers.A);
921 }
922
923 void CPU::LD_H_B()
924 {
925     LD_MM(_registers.H, _registers.B);
926 }
927
928 void CPU::LD_H_C()
929 {
930     LD_MM(_registers.H, _registers.C);
931 }
932
933 void CPU::LD_H_D()
934 {
935     LD_MM(_registers.H, _registers.D);
936 }
937
938 void CPU::LD_H_E()
939 {
940     LD_MM(_registers.H, _registers.E);
941 }
942
943 void CPU::LD_H_H()
944 {
945     LD_MM(_registers.H, _registers.H);
946 }
947
948 void CPU::LD_H_L()
949 {
950     LD_MM(_registers.H, _registers.L);
951 }
952
```

```
953 void CPU::LD_H_MM()
954 {
955 }
956
957 void CPU::LD_H_A()
958 {
959     LD_MM(_registers.H, _registers.A);
960 }
961
962 void CPU::LD_L_B()
963 {
964     LD_MM(_registers.L, _registers.B);
965 }
966
967 void CPU::LD_L_C()
968 {
969     LD_MM(_registers.L, _registers.C);
970 }
971
972 void CPU::LD_L_D()
973 {
974     LD_MM(_registers.L, _registers.D);
975 }
976
977 void CPU::LD_L_E()
978 {
979     LD_MM(_registers.L, _registers.E);
980 }
981
982 void CPU::LD_L_H()
983 {
984     LD_MM(_registers.L, _registers.H);
985 }
986
987 void CPU::LD_L_L()
988 {
989     LD_MM(_registers.L, _registers.L);
990 }
991
992 void CPU::LD_L_MM()
993 {
994 }
995
996 void CPU::LD_L_A()
997 {
998     LD_MM(_registers.L, _registers.A);
999 }
1000
1001 void CPU::LD_MM_B()
1002 {
1003 }
1004
1005 void CPU::LD_MM_C()
1006 {
1007 }
1008
```

```
1009 void CPU::LD_MM_D()
1010 {
1011 }
1012
1013 void CPU::LD_MM_E()
1014 {
1015 }
1016
1017 void CPU::LD_MM_H()
1018 {
1019 }
1020
1021 void CPU::LD_MM_L()
1022 {
1023 }
1024
1025 void CPU::HALT()
1026 {
1027 }
1028
1029 void CPU::LD_MM_A()
1030 {
1031 }
1032
1033 void CPU::LD_A_B()
1034 {
1035     LD_MM(_registers.A, _registers.B);
1036 }
1037
1038 void CPU::LD_A_C()
1039 {
1040     LD_MM(_registers.A, _registers.C);
1041 }
1042
1043 void CPU::LD_A_D()
1044 {
1045     LD_MM(_registers.A, _registers.D);
1046 }
1047
1048 void CPU::LD_A_E()
1049 {
1050     LD_MM(_registers.A, _registers.E);
1051 }
1052
1053 void CPU::LD_A_H()
1054 {
1055     LD_MM(_registers.A, _registers.H);
1056 }
1057
1058 void CPU::LD_A_L()
1059 {
1060     LD_MM(_registers.A, _registers.L);
1061 }
1062
1063 void CPU::LD_A_MM()
1064 {
```

```
1065 }
1066
1067 void CPU::LD_A_A()
1068 {
1069     LD_M_M(_registers.A, _registers.A);
1070 }
1071
1072 void CPU::ADD_A_B()
1073 {
1074     ADD_A_M(_registers.B);
1075 }
1076
1077 void CPU::ADD_A_C()
1078 {
1079     ADD_A_M(_registers.C);
1080 }
1081
1082 void CPU::ADD_A_D()
1083 {
1084     ADD_A_M(_registers.D);
1085 }
1086
1087 void CPU::ADD_A_E()
1088 {
1089     ADD_A_M(_registers.E);
1090 }
1091
1092 void CPU::ADD_A_H()
1093 {
1094     ADD_A_M(_registers.H);
1095 }
1096
1097 void CPU::ADD_A_L()
1098 {
1099     ADD_A_M(_registers.L);
1100 }
1101
1102 void CPU::ADD_A_MM()
1103 {
1104 }
1105
1106 void CPU::ADD_A_A()
1107 {
1108     ADD_A_M(_registers.A);
1109 }
1110
1111 void CPU::ADD_A_M(uint8 value)
1112 {
1113 }
1114
1115 void CPU::ADC_A_B()
1116 {
1117     ADC_A_M(_registers.B);
1118 }
1119
1120 void CPU::ADC_A_C()
```

```
1121 {
1122     ADC_A_M(_registers.C);
1123 }
1124
1125 void CPU::ADC_A_D()
1126 {
1127     ADC_A_M(_registers.D);
1128 }
1129
1130 void CPU::ADC_A_E()
1131 {
1132     ADC_A_M(_registers.E);
1133 }
1134
1135 void CPU::ADC_A_H()
1136 {
1137     ADC_A_M(_registers.H);
1138 }
1139
1140 void CPU::ADC_A_L()
1141 {
1142     ADC_A_M(_registers.L);
1143 }
1144
1145 void CPU::ADC_A_MM()
1146 {
1147 }
1148
1149 void CPU::ADC_A_A()
1150 {
1151     ADC_A_M(_registers.A);
1152 }
1153
1154 void CPU::ADC_A_M(uint8 value)
1155 {
1156 }
1157
1158 void CPU::SUB_A_B()
1159 {
1160     SUB_A_M(_registers.B);
1161 }
1162
1163 void CPU::SUB_A_C()
1164 {
1165     SUB_A_M(_registers.C);
1166 }
1167
1168 void CPU::SUB_A_D()
1169 {
1170     SUB_A_M(_registers.D);
1171 }
1172
1173 void CPU::SUB_A_E()
1174 {
1175     SUB_A_M(_registers.E);
1176 }
```

```
1177
1178 void CPU::SUB_A_H()
1179 {
1180     SUB_A_M(_registers.H);
1181 }
1182
1183 void CPU::SUB_A_L()
1184 {
1185     SUB_A_M(_registers.L);
1186 }
1187
1188 void CPU::SUB_A_MM()
1189 {
1190 }
1191
1192 void CPU::SUB_A_A()
1193 {
1194     SUB_A_M(_registers.A);
1195 }
1196
1197 void CPU::SUB_A_M(uint8 value)
1198 {
1199 }
1200
1201 void CPU::SBC_A_B()
1202 {
1203     SBC_A_M(_registers.B);
1204 }
1205
1206 void CPU::SBC_A_C()
1207 {
1208     SBC_A_M(_registers.C);
1209 }
1210
1211 void CPU::SBC_A_D()
1212 {
1213     SBC_A_M(_registers.D);
1214 }
1215
1216 void CPU::SBC_A_E()
1217 {
1218     SBC_A_M(_registers.E);
1219 }
1220
1221 void CPU::SBC_A_H()
1222 {
1223     SBC_A_M(_registers.H);
1224 }
1225
1226 void CPU::SBC_A_L()
1227 {
1228     SBC_A_M(_registers.L);
1229 }
1230
1231 void CPU::SBC_A_MM()
1232 {
```



```
1233 }
1234
1235 void CPU::SBC_A_A()
1236 {
1237     SBC_A_M(_registers.A);
1238 }
1239
1240 void CPU::SBC_A_M(uint8 value)
1241 {
1242 }
1243
1244 void CPU::AND_B()
1245 {
1246     AND_M(_registers.B);
1247 }
1248
1249 void CPU::AND_C()
1250 {
1251     AND_M(_registers.C);
1252 }
1253
1254 void CPU::AND_D()
1255 {
1256     AND_M(_registers.D);
1257 }
1258
1259 void CPU::AND_E()
1260 {
1261     AND_M(_registers.E);
1262 }
1263
1264 void CPU::AND_H()
1265 {
1266     AND_M(_registers.H);
1267 }
1268
1269 void CPU::AND_L()
1270 {
1271     AND_M(_registers.L);
1272 }
1273
1274 void CPU::AND_MM()
1275 {
1276 }
1277
1278 void CPU::AND_A()
1279 {
1280     AND_M(_registers.A);
1281 }
1282
1283 void CPU::AND_M(uint8 value)
1284 {
1285 }
1286
1287 void CPU::XOR_B()
1288 {
```

```
1289     XOR_M(_registers.B);
1290 }
1291
1292 void CPU::XOR_C()
1293 {
1294     XOR_M(_registers.C);
1295 }
1296
1297 void CPU::XOR_D()
1298 {
1299     XOR_M(_registers.D);
1300 }
1301
1302 void CPU::XOR_E()
1303 {
1304     XOR_M(_registers.E);
1305 }
1306
1307 void CPU::XOR_H()
1308 {
1309     XOR_M(_registers.H);
1310 }
1311
1312 void CPU::XOR_L()
1313 {
1314     XOR_M(_registers.L);
1315 }
1316
1317 void CPU::XOR_MM()
1318 {
1319 }
1320
1321 void CPU::XOR_A()
1322 {
1323     XOR_M(_registers.A);
1324 }
1325
1326 void CPU::XOR_M(uint8 value)
1327 {
1328 }
1329
1330 void CPU::OR_B()
1331 {
1332     OR_M(_registers.B);
1333 }
1334
1335 void CPU::OR_C()
1336 {
1337     OR_M(_registers.C);
1338 }
1339
1340 void CPU::OR_D()
1341 {
1342     OR_M(_registers.D);
1343 }
1344
```

```
1345 void CPU::OR_E()
1346 {
1347     OR_M(_registers.E);
1348 }
1349
1350 void CPU::OR_H()
1351 {
1352     OR_M(_registers.H);
1353 }
1354
1355 void CPU::OR_L()
1356 {
1357     OR_M(_registers.L);
1358 }
1359
1360 void CPU::OR_MM()
1361 {
1362 }
1363
1364 void CPU::OR_A()
1365 {
1366     OR_M(_registers.A);
1367 }
1368
1369 void CPU::OR_M(uint8 value)
1370 {
1371 }
1372
1373 void CPU::CP_B()
1374 {
1375     CP_M(_registers.B);
1376 }
1377
1378 void CPU::CP_C()
1379 {
1380     CP_M(_registers.C);
1381 }
1382
1383 void CPU::CP_D()
1384 {
1385     CP_M(_registers.D);
1386 }
1387
1388 void CPU::CP_E()
1389 {
1390     CP_M(_registers.E);
1391 }
1392
1393 void CPU::CP_H()
1394 {
1395     CP_M(_registers.H);
1396 }
1397
1398 void CPU::CP_L()
1399 {
1400     CP_M(_registers.L);
```

```
1401 }
1402
1403 void CPU::CP_MM()
1404 {
1405 }
1406
1407 void CPU::CP_A()
1408 {
1409     CP_M(_registers.A);
1410 }
1411
1412 void CPU::CP_M(uint8 value)
1413 {
1414 }
1415
1416 void CPU::RET_NZ()
1417 {
1418 }
1419
1420 void CPU::POP_BC()
1421 {
1422 }
1423
1424 void CPU::JP_NZ_RR()
1425 {
1426 }
1427
1428 void CPU::JP_RR()
1429 {
1430     uint16 jmpTo = Read16(_registers.PC);
1431     _registers.PC = jmpTo;
1432     _hasSetPC = TRUE;
1433 }
1434
1435 void CPU::CALL_NZ_RR()
1436 {
1437 }
1438
1439 void CPU::PUSH_BC()
1440 {
1441 }
1442
1443 void CPU::ADD_A_R()
1444 {
1445 }
1446
1447 void CPU::RST_00H()
1448 {
1449 }
1450
1451 void CPU::RET_Z()
1452 {
1453 }
1454
1455 void CPU::RET()
1456 {
```

```
1457 }
1458
1459 void CPU::JP_Z_RR()
1460 {
1461 }
1462
1463 void CPU::PREFIX_CB()
1464 {
1465 }
1466
1467 void CPU::CALL_Z_RR()
1468 {
1469 }
1470
1471 void CPU::CALL_RR()
1472 {
1473 }
1474
1475 void CPU::ADC_A_R()
1476 {
1477 }
1478
1479 void CPU::RST_08H()
1480 {
1481 }
1482
1483 void CPU::RET_NC()
1484 {
1485 }
1486
1487 void CPU::POP_DE()
1488 {
1489 }
1490
1491 void CPU::JP_NC_RR()
1492 {
1493 }
1494
1495 void CPU::CALL_NC_RR()
1496 {
1497 }
1498
1499 void CPU::PUSH_DE()
1500 {
1501 }
1502
1503 void CPU::SUB_R()
1504 {
1505 }
1506
1507 void CPU::RST_10H()
1508 {
1509 }
1510
1511 void CPU::RET_C()
1512 {
```

```
1513 }
1514
1515 void CPU::RETI()
1516 {
1517 }
1518
1519 void CPU::JP_C_RR()
1520 {
1521 }
1522
1523 void CPU::CALL_C_RR()
1524 {
1525 }
1526
1527 void CPU::SBC_A_R()
1528 {
1529 }
1530
1531 void CPU::RST_18H()
1532 {
1533 }
1534
1535 void CPU::LDH_R_A()
1536 {
1537 }
1538
1539 void CPU::POP_HL()
1540 {
1541 }
1542
1543 void CPU::LD_C_A2()
1544 {
1545 }
1546
1547 void CPU::PUSH_HL()
1548 {
1549 }
1550
1551 void CPU::AND_R()
1552 {
1553 }
1554
1555 void CPU::RST_20H()
1556 {
1557 }
1558
1559 void CPU::ADD_SP_R()
1560 {
1561 }
1562
1563 void CPU::JP_MM()
1564 {
1565 }
1566
1567 void CPU::LD_RR_A()
1568 {
```

```
1569 }
1570
1571 void CPU::XOR_R()
1572 {
1573 }
1574
1575 void CPU::RST_28H()
1576 {
1577 }
1578
1579 void CPU::LDH_A_R()
1580 {
1581     _registers.A = Read8(0xFF00 + Read8(_registers.PC));
1582     _registers.tClock.Cycle(3, 12);
1583 }
1584
1585 void CPU::POP_AF()
1586 {
1587 }
1588
1589 void CPU::LD_A_C2()
1590 {
1591     _registers.A = Read8(0xFF00 + _registers.C);
1592     _registers.tClock.Cycle(2, 8);
1593 }
1594
1595 void CPU::DI()
1596 {
1597 }
1598
1599 void CPU::PUSH_AF()
1600 {
1601 }
1602
1603 void CPU::OR_R()
1604 {
1605 }
1606
1607 void CPU::RST_30H()
1608 {
1609 }
1610
1611 void CPU::LD_HL_SPandR()
1612 {
1613 }
1614
1615 void CPU::LD_SP_HL()
1616 {
1617 }
1618
1619 void CPU::LD_A_RR()
1620 {
1621 }
1622
1623 void CPU::EI()
1624 {
```



```
1625 }
1626
1627 void CPU::CP_R()
1628 {
1629 }
1630
1631 void CPU::RST_38H()
1632 {
1633 }
1634
1635 void CPU::LD_M_M(uint8 & address, uint8 value)
1636 {
1637     address = value;
1638     _registers.tClock.Cycle(1, 4);
1639 }
1640
1641 void CPU::INC_M(uint8 & address)
1642 {
1643     address += 0x01;
1644     _registers.tClock.Cycle(1, 4);
1645 }
1646
1647 void CPU::DEC_M(uint8 & address)
1648 {
1649     address -= 0x01;
1650     _registers.tClock.Cycle(1, 4);
1651 }
1652
1653 void CPU::CB_RLC_B()
1654 {
1655     CB_RLC8(_registers.B);
1656 }
1657
1658 void CPU::CB_RLC_C()
1659 {
1660     CB_RLC8(_registers.C);
1661 }
1662
1663 void CPU::CB_RLC_D()
1664 {
1665     CB_RLC8(_registers.D);
1666 }
1667
1668 void CPU::CB_RLC_E()
1669 {
1670     CB_RLC8(_registers.E);
1671 }
1672
1673 void CPU::CB_RLC_H()
1674 {
1675     CB_RLC8(_registers.H);
1676 }
1677
1678 void CPU::CB_RLC_L()
1679 {
1680     CB_RLC8(_registers.L);
```

```
1681 }
1682
1683 void CPU::CB_RLC_MM()
1684 {
1685     _registers.tClock.Cycle(2, 16);
1686 }
1687
1688 void CPU::CB_RLC_A()
1689 {
1690     CB_RLC8(_registers.A);
1691 }
1692
1693 void CPU::CB_RRC_B()
1694 {
1695     CB_RRC8(_registers.B);
1696 }
1697
1698 void CPU::CB_RRC_C()
1699 {
1700     CB_RRC8(_registers.C);
1701 }
1702
1703 void CPU::CB_RRC_D()
1704 {
1705     CB_RRC8(_registers.D);
1706 }
1707
1708 void CPU::CB_RRC_E()
1709 {
1710     CB_RRC8(_registers.E);
1711 }
1712
1713 void CPU::CB_RRC_H()
1714 {
1715     CB_RRC8(_registers.H);
1716 }
1717
1718 void CPU::CB_RRC_L()
1719 {
1720     CB_RRC8(_registers.L);
1721 }
1722
1723 void CPU::CB_RRC_MM()
1724 {
1725     _registers.tClock.Cycle(2, 16);
1726 }
1727
1728 void CPU::CB_RRC_A()
1729 {
1730     CB_RRC8(_registers.A);
1731 }
1732
1733 void CPU::CB_RL_B()
1734 {
1735     CB_RL8(_registers.B);
1736 }
```

```
1737
1738 void CPU::CB_RL_C()
1739 {
1740     CB_RL8(_registers.C);
1741 }
1742
1743 void CPU::CB_RL_D()
1744 {
1745     CB_RL8(_registers.D);
1746 }
1747
1748 void CPU::CB_RL_E()
1749 {
1750     CB_RL8(_registers.E);
1751 }
1752
1753 void CPU::CB_RL_H()
1754 {
1755     CB_RL8(_registers.H);
1756 }
1757
1758 void CPU::CB_RL_L()
1759 {
1760     CB_RL8(_registers.L);
1761 }
1762
1763 void CPU::CB_RL_MM()
1764 {
1765     _registers.tClock.Cycle(2, 16);
1766 }
1767
1768 void CPU::CB_RL_A()
1769 {
1770     CB_RL8(_registers.A);
1771 }
1772
1773 void CPU::CB_RR_B()
1774 {
1775     CB_RR8(_registers.B);
1776 }
1777
1778 void CPU::CB_RR_C()
1779 {
1780     CB_RR8(_registers.C);
1781 }
1782
1783 void CPU::CB_RR_D()
1784 {
1785     CB_RR8(_registers.D);
1786 }
1787
1788 void CPU::CB_RR_E()
1789 {
1790     CB_RR8(_registers.E);
1791 }
1792
```

```
1793 void CPU::CB_RR_H()
1794 {
1795     CB_RR8(_registers.H);
1796 }
1797
1798 void CPU::CB_RR_L()
1799 {
1800     CB_RR8(_registers.L);
1801 }
1802
1803 void CPU::CB_RR_MM()
1804 {
1805     _registers.tClock.Cycle(2, 16);
1806 }
1807
1808 void CPU::CB_RR_A()
1809 {
1810     CB_RR8(_registers.A);
1811 }
1812
1813 void CPU::CB_SLA_B()
1814 {
1815     CB_SLA8(_registers.B);
1816 }
1817
1818 void CPU::CB_SLA_C()
1819 {
1820     CB_SLA8(_registers.C);
1821 }
1822
1823 void CPU::CB_SLA_D()
1824 {
1825     CB_SLA8(_registers.D);
1826 }
1827
1828 void CPU::CB_SLA_E()
1829 {
1830 }
1831
1832 void CPU::CB_SLA_H()
1833 {
1834     CB_SLA8(_registers.H);
1835 }
1836
1837 void CPU::CB_SLA_L()
1838 {
1839     CB_SLA8(_registers.L);
1840 }
1841
1842 void CPU::CB_SLA_MM()
1843 {
1844     _registers.tClock.Cycle(2, 16);
1845 }
1846
1847 void CPU::CB_SLA_A()
1848 {
```

```
1849     CB_SLA8(_registers.A);
1850 }
1851
1852 void CPU::CB_SRA_B()
1853 {
1854     CB_SRA8(_registers.B);
1855 }
1856
1857 void CPU::CB_SRA_C()
1858 {
1859     CB_SRA8(_registers.C);
1860 }
1861
1862 void CPU::CB_SRA_D()
1863 {
1864     CB_SRA8(_registers.D);
1865 }
1866
1867 void CPU::CB_SRA_E()
1868 {
1869     CB_SRA8(_registers.E);
1870 }
1871
1872 void CPU::CB_SRA_H()
1873 {
1874     CB_SRA8(_registers.H);
1875 }
1876
1877 void CPU::CB_SRA_L()
1878 {
1879     CB_SRA8(_registers.L);
1880 }
1881
1882 void CPU::CB_SRA_MM()
1883 {
1884     _registers.tClock.Cycle(2, 16);
1885 }
1886
1887 void CPU::CB_SRA_A()
1888 {
1889     CB_SRA8(_registers.A);
1890 }
1891
1892 void CPU::CB_SWAP_B()
1893 {
1894     CB_SWAP8(_registers.B);
1895 }
1896
1897 void CPU::CB_SWAP_C()
1898 {
1899     CB_SWAP8(_registers.C);
1900 }
1901
1902 void CPU::CB_SWAP_D()
1903 {
1904     CB_SWAP8(_registers.D);
```

```
1905 }
1906
1907 void CPU::CB_SWAP_E()
1908 {
1909     CB_SWAP8(_registers.E);
1910 }
1911
1912 void CPU::CB_SWAP_H()
1913 {
1914     CB_SWAP8(_registers.H);
1915 }
1916
1917 void CPU::CB_SWAP_L()
1918 {
1919     CB_SWAP8(_registers.L);
1920 }
1921
1922 void CPU::CB_SWAP_MM()
1923 {
1924     _registers.tClock.Cycle(2, 16);
1925 }
1926
1927 void CPU::CB_SWAP_A()
1928 {
1929     CB_SWAP8(_registers.A);
1930 }
1931
1932 void CPU::CB_SRL_B()
1933 {
1934     CB_SRL8(_registers.B);
1935 }
1936
1937 void CPU::CB_SRL_C()
1938 {
1939     CB_SRL8(_registers.C);
1940 }
1941
1942 void CPU::CB_SRL_D()
1943 {
1944     CB_SRL8(_registers.D);
1945 }
1946
1947 void CPU::CB_SRL_E()
1948 {
1949     CB_SRL8(_registers.E);
1950 }
1951
1952 void CPU::CB_SRL_H()
1953 {
1954     CB_SRL8(_registers.H);
1955 }
1956
1957 void CPU::CB_SRL_L()
1958 {
1959     CB_SRL8(_registers.L);
1960 }
```

```
1961
1962 void CPU::CB_SRL_MM()
1963 {
1964     _registers.tClock.Cycle(2, 16);
1965 }
1966
1967 void CPU::CB_SRL_A()
1968 {
1969     CB_SRL8(_registers.A);
1970 }
1971
1972 void CPU::CB_RLC8(uint8 & registerRef)
1973 {
1974     _registers.tClock.Cycle(2, 8);
1975 }
1976
1977 void CPU::CB_RRC8(uint8 & registerRef)
1978 {
1979     _registers.tClock.Cycle(2, 8);
1980 }
1981
1982 void CPU::CB_RL8(uint8 & registerRef)
1983 {
1984     _registers.tClock.Cycle(2, 8);
1985 }
1986
1987 void CPU::CB_RR8(uint8 & registerRef)
1988 {
1989     _registers.tClock.Cycle(2, 8);
1990 }
1991
1992 void CPU::CB_SLA8(uint8 & registerRef)
1993 {
1994     _registers.tClock.Cycle(2, 8);
1995 }
1996
1997 void CPU::CB_SRA8(uint8 & registerRef)
1998 {
1999     _registers.tClock.Cycle(2, 8);
2000 }
2001
2002 void CPU::CB_SWAP8(uint8 & registerRef)
2003 {
2004     _registers.tClock.Cycle(2, 8);
2005 }
2006
2007 void CPU::CB_SRL8(uint8 & registerRef)
2008 {
2009     _registers.tClock.Cycle(2, 8);
2010 }
2011
2012 void CPU::CB_BIT8(uint8 bit, uint8 & registerRef)
2013 {
2014     _registers.tClock.Cycle(2, 8);
2015 }
2016
```



```
2017 void CPU::CB_BIT16(uint8 bit, uint16 & registerRef)
2018 {
2019     _registers.tClock.Cycle(2, 16);
2020 }
2021
2022 void CPU::CB_RES8(uint8 bit, uint8 & registerRef)
2023 {
2024     _registers.tClock.Cycle(2, 8);
2025 }
2026
2027 void CPU::CB_RES16(uint8 bit, uint16 & registerRef)
2028 {
2029     _registers.tClock.Cycle(2, 16);
2030 }
2031
2032 void CPU::CB_SET8(uint8 bit, uint8 & registerRef)
2033 {
2034     _registers.tClock.Cycle(2, 8);
2035 }
2036
2037 void CPU::CB_SET16(uint8 bit, uint16 & registerRef)
2038 {
2039     _registers.tClock.Cycle(2, 16);
2040 }
2041
```