EECS 645 Computer Architecture

Homework 1

Download the homework file and extract the files.
Compile them using the included makefile and run the z80 executable.
It should run a simple test that results in output like the following:

```
./z80
PC: 0100, AF: 01b0, BC: 0804, DE: 0201, HL: 0000, SP: fffe
totalClockCycles at 00000000
instruction = 003c
 PC: 0101, AF: 02b0, BC: 0804, DE: 0201, HL: 0000, SP: fffe
totalClockCycles at 00000004
instruction = 003c
 PC: 0102, AF: 03b0, BC: 0804, DE: 0201, HL: 0000, SP: fffe
totalClockCycles at 00000008
instruction = 003c
 PC: 0103, AF: 04b0, BC: 0804, DE: 0201, HL: 0000, SP: fffe
totalClockCycles at 00000012
instruction = 0080
 PC: 0104, AF: 0cb0, BC: 0804, DE: 0201, HL: 0000, SP: fffe
totalClockCycles at 00000016
instruction = 0076
 PC: 0105, AF: 0cb0, BC: 0804, DE: 0201, HL: 0000, SP: fffe
totalClockCycles at 00000017
Total Clock Cycles: 17
Instruction 0x3c count is  0003
Instruction 0x76 count is  0001
Instruction 0x80 count is  0001
Halting now.
```

This verifies you have the example program working.

Implement the set of instructions listed below using the existing instructions in the decode case statement as a reference to give more capabilities to our Z80 processor.

You will find some macros defined that may be useful for byte manipulation.

Test your implementations by modifying the registers initial states, and adding your new instructions to the loadTest routine. Don’t forget to add the cycle counts to initMemory routine for each of your instructions.

Instructions to implement

| Instruction | Code | Cycles | Notes |
|---|---|---|---|
| Inc B | 04 | 4 | Increment the value in B, result in B |
| Dec B | 05 | 4 | Decrement the value in B, result in B |
| Inc C | 0C | 4 | |
| Dec C | 0D | 4 | |
| Inc D | 14 | 4 | |
| Dec D | 15 | 4 | |
| Inc E | 1C | 4 | |
| Dec E | 1D | 4 | |
| Inc H | 24 | 4 | |
| Dec H | 25 | 4 | |
| Inc L | 2C | 4 | |
| Dec L | 2D | 4 | |
| Inc A | 3C | 4 | Already done for you |
| Dec A | 3D | 4 | |
| And B | A0 | 4 | And B with A result in A |
| And C | A1 | 4 | |
| And D | A2 | 4 | |
| And E | A3 | 4 | |
| And H | A4 | 4 | |
| And L | A5 | 4 | |
| And A | A7 | 4 | And A with A result in A |
| Or B | B0 | 4 | Or B with A result in A |
| Or C | B1 | 4 | |
| Or D | B2 | 4 | |
| Or E | B3 | 4 | |
| Or H | B4 | 4 | |
| Or L | B5 | 4 | |
| Or A | B7 | 4 | Or A with A result in A |
| Xor B | A8 | 4 | Xor B with A result in A |
| Xor C | A9 | 4 | |
| Xor D | AA | 4 | |
| Xor E | AB | 4 | |
| Xor H | AC | 4 | |
| Xor L | AD | 4 | |
| Xor A | AF | 4 | Xor A with A result in A |
| Add B | 80 | 4 | Adds B to A result in A |
| Add C | 81 | 4 | |
| Add D | 82 | 4 | |
| Add E | 83 | 4 | |

| | | | |
|---|---|---|---|
| Add H | 84 | 4 | |
| Add L | 85 | 4 | |
| Add A | 87 | 4 | Adds A to A result in A |
| Sub B | 90 | 4 | Sub b from A result in A |
| Sub C | 91 | 4 | |
| Sub D | 92 | 4 | |
| Sub E | 93 | 4 | |
| Sub H | 94 | 4 | |
| Sub L | 95 | 4 | |
| Sub A | 97 | 4 | Sub A from A result in A |

When you have completed the implementation and testing, set the initial state of the registers back to this

```
// Registers
uint16_t AF = 0x01B0; // AAAAAAAAZNHCxxxx
uint16_t BC = 0x0804; // BBBBBBBBCCCCCCCC
uint16_t DE = 0x0201; // DDDDDDDDEEEEEEEE
uint16_t HL = 0x0000; // HHHHHHHHLLLLLLLL
uint16_t SP = 0xFFFE; // Stack Pointer
uint16_t PC = 0x0100; // Program Counter
```

```
Then modify the main loop and comment out loadTest, and uncomment
loadFirst.  Compile and run the program.  Place the program output
into a text file, then submit a zip/tar with your source code and the
text file output.
```