ESS: Exercise Set 1

Assembly Language

Use the following subset of assembler language for a fictious microcontroller. It is a *register* based 8-bit CPU, with byte registers b0...b31. It has two flags, Z and C. Z is set if the result of the last instruction was zero. C is set if the carry bit was set as a result of the last instruction.

Arithmetic Operations

ADD

Parameter: Byte-Register Byte-Register
Adds the values of both registers.
z: is set if the result is 0
c: is set if the result is greater than 255 (one byte)

ADD

Parameter: Byte-Register Byte-Literal
Adds the value of the register and the given literal value.
z: is set if the result is 0
c: is set if the result is greater than 255 (one word)

CMP

Parameter: Byte-Register Byte-Register
Compares two values.
z: is set if the values are equal
c: is set if the second value is greater than the first value

CMP

Parameter: Byte-Register Byte-Literal
Compares two values.
z: is set if the values are equal
c: is set if the second value is greater than the first value

DEC

Parameter: Byte-Register
Subtracts one from the value of the register. If the value is 0x00 the result will be 0xFF.
z: is set if the result is 0
c: is set if the value was 0

INC

Parameter: Byte-Register
Adds 1 to the value of the register. If the value is 0xFF, the result will be 0x00.
z: is set if the result is 0
c: is set if the value was 0xFF

Branch Operations

JMP

Parameter: Label
Jumps to the given label (unconditional jump)

JZ

Parameter: Label
Jumps to the given label only when the zero-flag is set.

JC

Parameter: Label
Jumps to the given label only when the carry-flag is set.

JNC

Parameter: Label
Jumps to the given label only when the carry-flag is not set.

JNZ

Parameter: Label
Jumps to the given label only when the zero-flag is not set.

Logical Operations

RR

Parameter: Byte-Register
Rotates the register by one bit to the right through the carry bit. If the carry-flag is set, the left most bit will be set. Example: 00000010 → 00000001
c: is set if the least-significant bit of the value was 1

RL

Parameter: Byte-Register
Shifts the register by one bit to the left into the carry bit. If the carry-flag is set, the right most bit will be set. Example: 00100000 → 01000000
c: is set if the most-significant bit of the value was 1

AND

Parameter: Byte-Register Byte-Register
Calculates the binary AND of the values of both registers.
z: is set if the result is 0

OR

Parameter: Byte-Register Byte-Register
Calculates the binary OR of the values of both registers.
z: is set if the result is 0

XOR

Parameter: Byte-Register Byte-Register
Calculates the binary exclusive-or of the values of both registers.
z: is set if the result is 0

INV

Parameter: Byte-Register
Calculates the 1's complement of the specified register.

z: is set if the result is 0

Register Operations

MOV
Parameter: Byte-Register Byte-Register
Copies the value of the second register into the first one.

MOV
Parameter: Byte-Register Byte-Literal
Writes the given literal value into the register.

Example Program

```
; Sample program
; Comments begin with a semi-colon

; Labels are a string, followed by a colon
INIT:
    ; Move the literal (constant) value 0x01 into register b0
    MOV b0, 0x01
    ; Move the contents of b0 to b1
    MOV b1, b0

; This is a new label
BLOB:
    ; We can refer to labels
    JMP BLOB
; We should always end with END, even if we never reach it
END
```

Question 1:

Implement a naive multiplier by repeated adding. Assume the one value to be multiplied is in register $b0$ and the other value is in register $b1$. The result should be stored in register $b2$. What should you be careful of?

Question 2:

Write a routine to count the number of bits that are set in a byte. For example 0x04 has one bit set, so the answer should be 1. Assume the input byte is in register $b0$ and the result should be stored in $b1$.

Question 3:

Write a routine to return the *nth* Fibonacci number, up to a maximum of the 10th number. The first Fibonacci number is 1, the second number is 1, the third number is 2, the fourth number is 3 and so on. Assume the number you want is in $b0$ and store the result in $b1$.

Question 4:

A *parity check* is a simple form of error checking that can detect single bit flips in a character. Parity bits (and more advanced cousins like Cyclical Redundancy Checks) are used in data transmission to check message integrity. Parity determines whether or not a byte has an even number of 1's or an odd number of 1's. Given a byte in register $b0$, indicate in register $b1$ whether it is even-parity or odd-parity.