GIFT School of Engineering and Applied Sciences

Spring 2022

CS-345: Computer organization and assembly language

Lab-5 Manual

Registers and Flags

REGISTERS AND FLAGS

OBJECTIVES

1. How to interpret the different types of registers available in the IAPX86 architecture.
2. How to use the different types of registers and how to manipulate them in assembly language.
3. How to interpret the function of flags and to use them effectively.
4. How to perform arithmetic operations with registers.
5. How to use the debugger for viewing the available registers and their function.

THEORY

The 8086 chip uses registers for performing operations. Following is a brief detail of their functions:

General registers: There are four 16-bit general-purpose registers used primarily to contain operands for arithmetic and logical operations.

Segment registers: These special-purpose registers permit systems software designers to choose either a flat or segmented model of memory organization. These four registers determine, at any given time, which segments of memory are currently addressable.

Status and instruction registers: These special-purpose registers are used to record and alter certain aspects of the 8086 processor state.

Flags Register: FLAGS is a 16-bit register which should be interpreted as a collection of single bit flags rather than as a unit. The flags may be considered in two major groups: the status flags and the control flags.

Status Flags: The status flags of the FLAGS register allow the results of one instruction to influence later instructions. The arithmetic instructions use OF, SF, ZF, AF, PF, and CF. The SCAS (Scan String), CMPS (Compare String), and LOOP instructions use ZF to signal that their operations are complete. There are instructions to set, clear, and complement CF before execution of an arithmetic instruction.

Control Flags: The control flag DF of the EFLAGS register controls string instructions. Setting DF causes string instructions to auto-decrement; that is, to process strings from high addresses to low addresses. Clearing DF causes string instructions to auto-increment, or to process strings from low addresses to high addresses.

Instruction Pointer: The instruction pointer register (IP) contains the offset address, relative to the start of the current code segment, of the next sequential instruction to be executed. The instruction pointer is not directly visible to the programmer; it is controlled implicitly by control-transfer instructions, interrupts and exceptions. Control flags are given below:

Segment Registers		
CS	Code Segment	16-bit number that points to the active code-segment
DS	Data Segment	16-bit number that points to the active data-segment
SS	Stack Segment	16-bit number that points to the active stack-segment
ES	Extra Segment	16-bit number that points to the active extra-segment
Pointer Registers		
IP	Instruction Pointer	16-bit number that points to the offset of the next instruction
SP	Stack Pointer	16-bit number that points to the offset that the stack is using
BP	Base Pointer	used to pass data to and from the stack
General-Purpose Registers		
AX	Accumulator Register	mostly used for calculations and for input/output
BX	Base Register	Only register that can be used as an index
CX	Count Register	register used for the loop instruction
DX	Data Register	input/output and used for multiply and divide
Index Registers		
SI	Source Index	used by string operations as source
DI	Destination Index	used by string operations as destination

Summary of Registers

.

Abr.	Name	bit n°	Description
OF	Overflow Flag	11	indicates an overflow when set
DF	Direction Flag	10	used for string operations to check direction
IF	Interrupt Flag	9	if set, interrupt are enabled, else disabled
TF	Trap Flag	8	if set, CPU can work in single step mode
SF	Sign Flag	7	if set, resulting number of calculation is negative
ZF	Zero Flag	6	if set, resulting number of calculation is zero
AF	Auxiliary Carry	4	some sort of second carry flag

PROBLEMS

1. Give the value of the IP, zero flag, the carry flag, the sign flag, and the overflow flag after each of the following instructions if AX is initialized with 0x1254 and BX is initialized with 0x0FFF.

```
add ax, 0xEDAB
add ax, bx
add bx, 0xF001
```

2. Write instructions that perform the following operations.
   a. Copy BL into CL
   b. Copy DX into AX
   c. Store 0x12 into AL
   d. Store 0x1234 into AX
   e. Store 0xFFFF into AX

3. Write the program following programs (from class source) using only CX and DX registers, and trace them line by line.

   a. `p1.asm`
   b. `p2.asm`
   c. `p3.asm`

4. Assemble program p3.asm without declaring label.

5. Write a program to write 0x3A, 0x2B and 0x3C on addresses 0x0117, 0x0118 and 0x0119 without reserving them. Also discuss the potential issues of this program.

REFERENCES

1. The Art Of Assembly Language Programming, http://webster.cs.ucr.edu/Page_win32/
2. Intel 80386 Reference Programmer's Manual, http://www.logix.cz/michal/doc/i386/
3. Assembly Language Programming Book by Belal Mohammad Hashmi, NUCES FAST, Lahore, Pakistan
4. Assembly Tutorial, http://www.geocities.com/SiliconValley/6112/asm0100.htm