Lecture Title

Course Code: 0052

Course Title: : Computer Organization and Architecture

Dept. of Computer Science
Faculty of Science and Technology

Lecturer No:	**7**	**Week No:**	**7**	**Semester:**	
Lecturer:	*Name & email*				

Overview : LOGIC

- Instructions to **change the bit pattern** in a byte or word
- The ability to **manipulate bits manually** which is unlikely in high level languages (Except C)
- **Logic Instructions:** AND, OR, XOR and NOT
- Logic Instructions can be used to **clear, set, and examine** bits, a register or variable. i.e. these will be used for
    - Converting a lowercase letter to upper case
    - Determining If a register contains an even or odd number.

LOGIC

- Instructions to change the bit pattern in a byte or word
- The ability to manipulate bits manually which is unlikely in high level languages (Except C)
- Logic Instructions: AND, OR, XOR and NOT
- Logic Instructions can be used to clear, set, and examine bits, a register or variable. i.e. these will be used for
- Converting a lowercase letter to upper case
- Determining If a register contains an even or odd number.

LOGIC Instructions

- The ability to manipulate individual bits is one the main advantages of assembly language.
- Individual bits can be changed in computer by using logic operations.
- The binary values of 0 = False and 1= True
- When a logic operation is applied to 8- or 16-bit operands, the result is obtained by applying the logic operation at each bit position.

AND, OR, and XOR instructions

- The AND, OR, and XOR instructions perform the named logic operations. The formats are:
  - AND **destination**, source
  - OR **destination**, source
  - XOR **destination**, source
- The result of the operation is stored in the destination, which must be a register or memory location.
- The source may be a constant, register, or memory location.
- However, memory-to-memory operations are not allowed.

Effect on Flags

- SF, ZF, PF reflect the result
- AF is undefined
- CF, OF= 0

- One use of AND, OR, and XOR is to selectively modify the bits in the destination.
- To do this, we construct a source bit pattern known as mask.
- The mask bits are chosen so that the corresponding destination bits are modified in the desired manner when the instruction is executed.

MASK

- To choose the mask bits, we make use of the following properties of AND, OR, and XOR:
- The **AND** instruction can be used to **CLEAR** specific destination bits while preserving the others.
  - A **0 mask** bit **clears** the corresponding destination bit.
  - a **1 mask** bit **preserves** the corresponding destination bit.
- The **OR** instruction can be used to **SET** specific destination bits while preserving the others.
  - A **1 mask** bit **sets** the corresponding destination bit.
  - A **0 mask** bit **preserves** the corresponding destination bit. ·
- The **XOR** instruction can be used to **complement** specific destination bits while preserving the others.
  - A **1 mask** bit **complements** the corresponding destination bit;
  - A **0 mask** bit preserves the corresponding destination bit.

Not Instruction

- The NOT instruction performs the **one's complement** operation on the destination. The format is:
  - **NOT destination** (**No effect on status flags)
  - Example: Complement the bits in AX:
    - **NOT AX**

TEST Instruction

- The **TEST** Instruction performs an AND operation of the destination with the source but **does not change** the destination contents.
- The purpose of the test instruction is to **set the status flags**. The format is:
  - TEST destination, Source
- Effects of flags on test operation:
  - CF, OF =0
  - AF = Undefined
  - SF, ZF, PF reflect the result

Bit Examination on TEST

- TEST instruction can be used to examine individual bits in operand.
- The mask should contain **1's** in the bit positions to be tested and **0's** elsewhere
    - As **1 AND b = b**, **0 AND b = 0**
- The operation **TEST destination, mask**
- Will have 1's in the tested bit positions if and only if the destination has 1's in these positions; and 0's elsewhere.
- if the destination has 0's in all the tested positions, the result will be 0 and thus ZF=1

References

- Assembly Language Programming and Organization of the IBM PC, Ytha Yu and Charles Marut, McGraw Hill, 1992. (ISBN: 0-07-072692-2).
- http://faculty.cs.niu.edu/~byrnes/csci360/notes/360shift.htm

Books

- Assembly Language Programming and Organization of the IBM PC, Ytha Yu and Charles Marut, McGraw Hill, 1992. (ISBN: 0-07-072692-2).

- Essentials of Computer Organization and Architecture, (Third Edition), Linda Null and Julia Lobur

- W. Stallings, “Computer Organization and Architecture: Designing for performance”, 67h Edition, Prentice Hall of India, 2003, ISBN 81 – 203 – 2962 – 7

- Computer Organization and Architecture by John P. Haynes.