NIMRA IQBAL

LAB 2

BASIC REGISTERS AND DATA DECLARATIONS

Registers

The basic purpose of a computer is to perform operations and operations need operands.

For example addition operation

- It involves adding two numbers
- We can have one precisely one address on the address bus and consequently precisely one element on data bus.
- At the same instant the second operand cannot brought inside the processor.
- As soon as the second is selected, the first operand is no longer there.
- For this reason there are temporary storage places inside the processor called REGISTERS

General Instruction Format

Instruction dest,src

Instruction dest

Instruction src(implied operand)

ACCUMULATOR

- There is a central register in every processor called the accumulator.
- Traditionally all mathematical and logical operations are performed on the accumulator.
- A 32 bit processor has an accumulator of 32 bit.

Pointer/Index/Base

- It holds the address of operands.

Flag Registers or Program Status Word:

- This is a special register in every architecture called the flag registers or program status word.
- Like the accumulator it is meaningless as a unit, rather the individual bit carry different meanings.
- The bit of accumulator work in a parallel as a unit and each bit mean the same thing.
- The bits of the flags register work independently and individually, and combined its value meaningless.

Program Counter or Instruction Pointer:

- The program counter holds the address of the next instruction to be executed.

Instruction Group

- Data Movement Instructions
- Arithemetic/Logical Instructions
- Program Control Instructions
- Special Instructions

Data Movement Instructions

- mov ax,bx ;move data from bx to ax

Arthimetic/Logic Instructions:

- add bx, 53 ;add 53 with bx
- and ax,123 ;and 123 to ax

A simple program Explanation

Move 5 to AX mov ax,5

Move 10 to BX mov bx,10

Add BX to AX add ax,bx

Move 15 to BX mov bx,15

Add BX to AX add ax,bx

The first program we are going to run is below. Copy this into theproject. Run the program.

Let’s talk about what this program does.

```
 TITLE Add and Subtract ;
INCLUDE Irvine32.inc
.code
main PROC
mov eax,10000h
add eax,40000h
sub eax,20000h
call DumpRegs
 exit main ENDP
 END main
```

Program Control: (Discuss in more details in further labs)

- Few examples
- cmp 0,ax ;compare ax with 0
- jne 123 ;Jump if not equal to the instruction at 123

Special Instructions: (Discuss in more details in further labs)

- cli ;clear the interrupt flag
- sti ;set the interrupt flag

Register Architecture: (16BIT)

- General Purpose Register:
- AX A stands for accumulator
- BX B stands for Base
- CX C stands for counter
- DX D stands for destination

- These registers can also be accessible as pairs of 8 bits.
- AX (AH and AL)
- BX (BH and BL)
- CX (CH and CL)
- DX (DH and DL)

INDEX POINTER/BASE

- SI source Index
- DI Destination index

Instruction Pointer

- IP contains the address of next pointer

Stack Pointer

SP (will be explored in the discussion of stack system)

Base Pointer

A memory pointer containing the address in a special area of memory called the stack.

Flag Registers

- The individual flags are discussed as under:
- C(Carry Flag): Carry flags is for the carry from the whole addition.
- P(Parity Flag): Verify the integrity of data sent from the sender to the receiver
- A(Auxiliary Flag): Auxiliary carry is the carry from the first nibble to the second.
- Z (Zero Flag): The zero flag is set if the last mathematical or logical instruction has produced a zero in its destination.
- S(sign Flag): The sign bit of the last mathematical or logical operations's destination is copied into the sign bit.
- T (Trap Flag) will discuss later
- I (Interrupt flag)
- D (Direction Flag)
- O (Overflow flag)

Example:

What will be the value of the flag after the following instruction sequences has executed?

- TITLE My First Program (Test.asm)
- INCLUDE Irvine32.inc
- .code
- main PROC
- mov ax,0FFFFh
- add ax,01h
- call DumpRegs
- exit
- main ENDP
- END main

Word representation

1. Little endian (Intel)
2. Big Endian (Motorolla)

Lets talk about data type ,data declaration (.data segment)

Data Type

- BYTE 8-bit unsigned integer
- SBYTE 8-bit signed integer
- WORD 16-bit unsigned integer
- SWORD 16-bit signed integer
- DWORD 32-bit unsigned integer
- SDWORD 32-bit signed integer
- FWORD 48-bit integer (Far pointer in protected mode)
- QWORD 64-bit integer
- TBYTE 80-bit (10-byte) integer
- REAL4 32-bit (4-byte) short real
- REAL8 64-bit (8-byte) long real
- REAL10 80-bit (10-byte) extended real

Defining BYTE and SBYTE Data

- • Each of following defines a single byte of storage:
- **value1 BYTE 'A' ; character constant**
- **value2 BYTE 0 ; smallest unsigned byte**
- **value3 BYTE 255 ; largest unsigned byte**
- **value4 SBYTE -128 ; smallest signed byte**
- **value5 SBYTE +127 ; largest signed byte**
- **value6 BYTE ? ; uninitialized byte**

Defining Byte Array:

Examples that use multiple initializers:

- **list1 BYTE 10,20,30,40**
- **list2 BYTE 10,20,30,40**
- **BYTE 50,60,70,80**
- **BYTE 81,82,83,84**
- **list3 BYTE ?,32,41h,00100010b**
- **list4 BYTE 0Ah,20h,‘A’,22h**

Defining Strings

• An array of characters

• Usually enclosed in quotation marks

• Will often be null-terminated

• To continue a single string across multiple lines, end each line with a comma

- **str1 BYTE "Enter your name",0**
- **str2 BYTE 'Error: halting program',0**
- **str3 BYTE 'A','E','I','O','U'**
- **greeting BYTE "Welcome to the Encryption Demo program "**
- **BYTE "created by Kip Irvine.",0**
- **menu BYTE "Checking Account",0dh,0ah,0dh,0ah,**
- **"1. Create a new account",0dh,0ah,**
- **"2. Open an existing account",0dh,0ah,**
- **"Choice> ",0**

End-of-line sequence:

- 0Dh = carriage return
- 0Ah = line feed

Using the DUP Operator

Use DUP to allocate (create space for) an array or string

Syntax:

- **counter DUP (argument)**

Counter and argument must be constants or constant expressions

- **var1 BYTE 20 DUP(0) ; 20 bytes, all equal to zero**
- **var2 BYTE 20 DUP(?) ; 20 bytes, uninitialized**
- **var3 BYTE 4 DUP("STACK"); 20 bytes, "STACKSTACKSTACKSTACK”**

Defining WORD and SWORD

- Define storage for 16-bit integers
- or double characters
- single value or multiple values
- **word1 WORD 65535 ; largest unsigned value**
- **word2 SWORD –32768 ; smallest signed value**
- **word3 WORD ? ; uninitialized, unsigned**
- **word4 WORD "AB" ; double characters**
- **myList WORD 1,2,3,4,5 ; array of words**
- **array WORD 5 DUP(?) ; uninitialized array**

Defining Other Types of Data

Storage definitions for 32-bit integers, quadwords, tenbyte values, and real numbers:

- **val1 DWORD 12345678h ; unsigned**
- **val2 SDWORD –2147483648 ; signed**
- **val3 DWORD 20 DUP(?) ; unsigned array**
- **val4 SDWORD –3,–2,–1,0,1 ; signed array**
- **quad1 QWORD 1234567812345678h**
- **val1 TBYTE 10000000000123456789Ah**

EXERCISE TIME