Arrays and Addressing Modes

Outline

- One-dimensional array
- Addressing modes
- An application of sorting an array
- Two-dimensional array
- Based indexed addressing mode
- The XLAT instruction

Arrays

- It is necessary to treat a collection of values as a group.

- The advantage of using an array to store the data is that a single name can be given to the whole structure and an element can be accessed by providing an index.

One-dimensional Array

- A one-dimensional array is an ordered list of elements all of the same type.
- By "ordered“, we mean that there is a first element, second element, third element and so on.
- In mathematics, if A is an array, the elements are usually denoted by A[1], A[2], A[3], and so on.
- Byte array declaration:
  - MSG DB 'abcde’
- Word array declaration:
  - w DW 10, 20, 30, 40, 50, 60
- The address of the array variable is called the base address of the array.

Two-dimensional Array

- A two-dimensional array is an array of arrays.
- We can picture the elements as being arranged in rows and columns.
- Because memory is one-dimensional, the elements of a two-dimensional array must be stored sequentially.
- There are two commonly used ways:
  - Row-major order.
  - Column-major order.

The DUP Operator

- It is possible to define arrays whose elements share a common initial value by using the DUP (duplicate) operator.
- Syntax:
  - repeat_count DUP (value)
- This operator causes value to be repeated the number of times specified by repeat_count.
- Example
  - GAMMA DW 100 DUP (0)
- DUPs may be nested. For example,
  - LINE DB 5, 4, 3 DUP (2, 3 DUP (0) , 1) which is equivalent to

    LINE DB 5, 4, 2, 0, 0, 0, 1, 2, 0, 0, 0, 1, 2, 0, 0, 0, 1

Location of Array Elements

- The address of an array element may be specified by adding a constant to the base address.
- Suppose A is an array and S denotes the number of bytes in an element.
- S = 1 for a byte array, S =2 for a word array.
- The position of the elements in array A can be determined as A=(N-1)x S.

Addressing Modes

- The way an operand is specified is known as its addressing mode.
- The addressing modes we have used so far are:
  1. Register mode, which means that an operand is a register
  2. Immediate mode, when an operand is a constant.
  3. Direct mode, when an operand is a variable.
- There are four additional addressing modes for the 8086:
  1. Register Indirect mode.
  2. Based mode.
  3. Indexed mode.
  4. Based Indexed mode.
- These modes are used to address memory operands indirectly.

Register Indirect Mode

- In this mode, the offset address of the operand is contained in a register.
- The register acts as a pointer to the memory location.
- The operand format is [register].
- The register is BX, SI, DI or BP.
- For BX, SI or DI the operand's segment number is contained in DS.
- For BP, SS has the segment number.

Register Indirect Mode

- For example, suppose that SI contains 0100h and the word at 0100h contains 1234h. To execute MOV AX, [SI] the CPU
  1. examines SI and obtains the offset address 0100h.
  2. uses the address DS:0100h to obtain the value 1234h.
  3. moves 1234h to AX.

Based Addressing Mode

- In this mode, the operand's offset address is obtained by adding a number called a displacement to the contents of a register.
- Displacement may be any of the following:
  - The offset address of a variable.
  - A constant (positive or negative).
  - The offset address of a variable plus or minus a constant.
- The syntax of an operand is any of the following equivalent expressions:
  - [register + displacement], [displacement + register], [register] + displacement , displacement + [register], displacement[register]
- The register must be BX or BP. If BX is used, DS contains the segment number of the operand's address. If BP is used, SS has the segment number.

Indexed Addressing Mode

- In this mode, the operand's offset address is obtained by adding a number called a displacement to the contents of a register.
- Displacement may be any of the following:
  - The offset address of a variable.
  - A constant (positive or negative).
  - The offset address of a variable plus or minus a constant.
- The syntax of an operand is any of the following equivalent expressions:
  - [register + displacement], [displacement + register], [register] + displacement , displacement + [register], displacement[register]
- The register must be SI or Dl. If SI or DI is used, DS contains the segment number of the operand's address.

Based Indexed Addressing Mode

- In this mode,. the offset address of the operand is the sum of
  1. The contents of a base register (BX or BP).
  2. The contents of an index register (SI or DI).
  3. Optionally, a variable's offset address.
  4. Optionally, a constant (positive or negative).
- If BX is used, DS contains the segment number of the operand's address.
- If BP is used, SS has the segment number.
- The operand may be written several ways. Four of them are
  1. variable[base_register] [index_register]
  2. [base_register + index_register + variable + constant]
  3. variable[base_register + index_register + constant]
  4. constant[base_register + index_register + variable]

The PTR Operator

- Assembler can't assemble MOV [BX], 1.
- Because it can’t tell whether the destination is the byte pointed to by BX or the word pointed to by BX.
- If the destination is supposed to be a byte, we can write
  - MOV BYTE PTR [BX] , 1
- If the destination is supposed to be a word, we can write
  - MOV WORD PTR [BX] , 1
- Syntax:
  - type PTR address_expression

The LABEL Pseudo-op

- Using LABEL pseudo-op code we can solve the type conflict.
- Example:

  -
    ```
    MONEY   LABEL  WORD
    DOLLARS  DB   1AH
    CENTS   DB   52H
    ```
- This declaration types MONEY as a word variable, and the components DOLLARS and CENTS as byte variables, with MONEY and DOLLARS being assigned the same address by the assembler.

  -
    ```
    MOV   AX,   MONEY          ; AL  =  dollars,    AH  =  cents
    ```

The XLAT Instruction

- The instruction XLAT (translate) is a no-operand instruction that can be used to convert a byte value into another value that comes from a table.
- The byte to be converted must be in AL, and BX has the offset address of the conversion table.
- The instruction
  1. Adds the contents of AL to the address in BX to produce an address within the table.
  2. Replaces the contents of AL by the value found at that address.