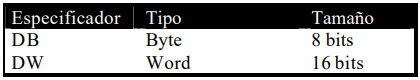
**Nombre\_variable especificador\_tipo valor\_inicial**

**?:** El valor inicial no se especifica usando este carácter



**h**: hexadecimal cuando termina el número, en caso de no tener, es un decimal

**nombre EQU valor** definimos constantes

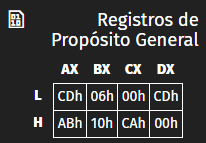
**nombre\_variable especificador\_tipo valores** definir variables

**tabla DB**: 1, 2, 4, 8, 16, 32, 64, 128

**string DB “Esto es un String”**

**cantidad DUP (Valores)** cantidad indica la cantidad de veces que se repite

**ORG dirección** Permite cambiar la dirección una vez que el programa se ejecuto



Los podemos tratar como registros de 16 bit o como un par de registros de 8 bits, tomando la parte baja separada de la alta. (L: low o bajo y H: high o alto)

**BL** Registro en bajo; **BX** Registro en alto

**MOV destino, origen** Abreviatura de mover MOV BL, 1 == BL=1

**ADD (Sumar) SUB (Restar)**

**ADD: operando1, operando2 (operando1 + operando2)**

**SUB: operando1, operando2 (operando1 - operando2)**

**ADC: operando1, operando2 (operando1 + operando2) + C (carry)**

**SBB: operando1, operando2 (operando1 - operando2) - C (carry)**

**INC: operando 🡨 operando + 1 Incrementar**

**DEC: operando 🡨 operando – 1 Decrementar**

**AND operando1, operando2**

**OR operando1, operando2**

**XOR operando1, operando2**

**NOT operando**

**NEG operando**

**CMP Compare: Comparar**

**JMP dirección ; Salta siempre Jump = Salto en ingles**

**JZ dirección ; Salta si el flag Z=1**

**JNZ dirección ; Salta si el flag Z=0**

**JS dirección ; Salta si el flag S=1**

**JNS dirección ; Salta si el flag S=0**

**JC dirección ; Salta si el flag C=1**

**JNC dirección ; Salta si el flag C=0**

**JO dirección ; Salta si el flag O=1**

**JNO dirección ; Salta si el flag O=0**

**HLT Halt: detener en inglés (Detiene el cpu)**

**NOP No hace nada**

**For**

**AL := 0;**

**FOR CL := 1 TO 10 DO**

**AL := AL + AL;**

**MOV AL, 0**

**MOV CL, 1**

**Iterar: CMP CL, 10**

**JZ Fin**

**ADD AL, AL**

**INC CL**

**JMP Iterar**

**Fin: HLT**

**For downto**

**MOV AL, 0**

**MOV CL, 10**

**Iterar: CMP CL, 1**

**JZ Fin**

**ADD AL, AL**

**DEC CL**

**JMP Iterar**

**Fin: HLT**

**Repeat until**

**MOV AL, 0**

**MOV CL, 10**

**Iterar: ADD AL, AL**

**DEC CL**

**CMP CL, 1**

**JNZ Iterar**

**Fin: HLT**

**Resumiendo, todas las estructuras de iteración se resuelven de la misma manera. Lo que varía es donde y que condición se evalúa.**

**Encontrar un máximo**

**const**

**tabla: array[1..10] of Word = {5, 2, 10, 4, 5, 0, 4, 8, 1, 9};**

**var**

**max: Word;**

**begin**

**max := 0;**

**for i := 1 to 10 do**

**if tabla[i] > max then**

**max := tabla[i];**

**end.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**ORG 1000h**

**tabla dw 5, 2, 10, 4, 5, 0, 4, 8, 1, 9;**

**max dw 0**

**ORG 2000h**

**MOV BX, OFFSET tabla ; (1)**

**MOV CL, 0 ; (2)**

**MOV AX, max ; (3)**

**Loop: CMP [BX], AX ; (4)**

**JC Menor ; (5)**

**MOV AX, [BX] ; (6)**

**Menor: ADD BX, 2 ; (7)**

**INC CL ; (8)**

**CMP CL, 10 ; (9)**

**JNZ Loop ; (10)**

**MOV max, AX ; (11)**

**HLT ; (12)**

**END**