FORMATO DO PROGRAMA FONTE

As declarações do programa fonte são constituídas pelos seguintes campos:

<Rótulo> <Operação> <Operando> <Comentário>

1) Campo do Rótulo:

- o primeiro caractere deve ser alfabético e pode ter no máximo 13 caracteres
- espaço, "tab" e " : " são considerados como caracteres finais do Rótulo
- corresponde ao endereço da instrução
- é opcional
- para identação do programa usar “tab” antes do próximo campo
- alinhar o primeiro caractere do Rótulo à esquerda

2) Campo da Operação :

- contém o mnemônico da instrução ou diretivas do programa,
- não diferencia entre maiúsculas e minúsculas.

3) Campo do Operando:

- especifica o dado a ser operado pela instrução.

4) Campo do Comentário:

- Usado pelo programador para comentar a função da instrução no contexto do programa.
- É opcional.
- Sempre começa com ";" .
- Se o comentário mudar de linha, deve vir precedido de “ ; ”

Exemplo:

```
INIT:    Mov A,30h ; carrega o acumulador com o dado
                   ;armazenado no endereço 30h.
```

TIPOS DE INFORMAÇÕES NO CAMPO DO OPERANDO

1. **Rótulo** - é um conjunto de caracteres com valor numérico associado a ele, e geralmente representando um endereço. Pode ter no máximo 13 caracteres, sendo o primeiro obrigatoriamente uma letra . Os demais caracteres podem ser letras, dígitos e ponto.

2. **Constante numérica –**

   - **Decimal** - é o default; o final D é opcional.

Exemplo:
ABC: MOV A,#15D ; carrega o registrador A com 15 decimal

   - **Hexadecimal** - a constante deve ser finalizada com H; quando inicia com uma letra deve ser precedida por 0(zero) .

Exemplo :
Here: Mov A, #0BAH ; carrega o registrador A com BAh

   - **Octal** - deve ser finalizada com Q

Exemplo:
La12: MOV A,#72Q ; carrega o acumulador com 72 octal

   - **Binária** - deve ser finalizada com B.

Exemplo:
dda1: MOV A, #11110110B ; carrega o registrador A com F6H

   - **Caracteres ASCII -** A constante ASCII deve vir entre cotas únicas.

Exemplo:
M1: Mov A, #`E` ; carrega o registrador A com o código ASCII da letra E

   - **Contador de posição** - o valor corrente do PC pode ser usado em expressões colocando-se um $ na posição desejada da expressão..

Exemplo:
kk: sjmp kk ;é equivalente a
kk: sjmp $

PSEUDO-INSTRUÇÕES OU DIRETIVAS DO ASSEMBLER

As diretivas não geram código de máquina!!.

São utilizadas no programa para complementar as informações que permitam a montagem efetiva do programa tais como: indicar endereço inicial do programa, reservar área de dados, definir equivalência entre identificadores e valores.

O programa PINNACLE para o microcontrolador da família MCS-51 aceita as seguintes diretivas (ver help diretamente no menu do programa):

Assemby Language Directives
__PINNACLE__ - Predefined symbol
INCLUDE - Include an external file in assembly
= - Equate symbol assignment
BIT - Bit symbol assignment
DB - Data Byte directive
DEFINE - Define symbol assignment
DS - Reserve Data Space in current segment
DW -Data Word directive
ELSE - ELSE conditional assembly
ENDIF - ENDIF conditional assembly
ENDMAC - Ends a macro declaration
EQU - Equate symbol assignment
EXTERN - Declare External symbol
IF - IF conditional assembly
IFDEF - If Defined conditional assembly
IFNDEF - If Not Defined conditional assembly
LIBRARY - Library module declaration
MACRO - Initiate a macro declaration
ORG - Set new assembly address
PROGRAM - Program module declaration
PUBLIC - Declare symbol as public
SET - Variable symbol assignment

Principais diretivas:

1) Diretiva ORG – define a Origem do programa

ORG *endereço*

A diretiva ORG deve ser usada para instruir ao Assembler em qual endereço deve começar a colocar o código do programa compilado.
Por default, na ausência da diretiva ORG, o código do programa começa no endereço 0000h, que é o endereço de reset dos microcontroladores da família MCS-51.

O valor do endereço deve ser uma expressão válida. Ou seja, o endereço pode ser um valor numérico válido ou conter uma expressão com contador de posição.

Por exemplo:

```
ORG 0 ; inicia o código do programa no endereço zero (endereço de reset do microcontrolador)

ORG 10h ; inicia o código do programa no endereço 10 hexadecimal.

ORG $ + 10h  ; Inicia o código do programa 10h posições acima do endereço onde está
             ; localizada a diretiva ORG. O símbolo $ é substituído pelo endereço
             ; corrente. Se o endereço corrente é  0200h, a linha de comando acima faz
             ; com que o código do programa comece no endereço 0210h.
```

2) Diretiva DB – Define Byte

DB databyte1 *[, databyte2, [databyte3...]]*
DB "string1" *[, “string2” [, “string3”...]]*

A diretiva DB permite ao programador inserir bytes de dados diretamente no programa na posição de memória corrente.
Os valores numéricos de 8 Bits são inseridos respeitando-se o seu formato (decimal, hexadecimal, binário, octal). Se mais de um valor for inserido eles devem vir separados por vírgula.
Caracteres ASCII isolados ou “Strings” de caracteres ASCII devem estar contidos entre aspas.
Obs: Esta diretiva deve ser colocada sempre depois do fim lógico do programa para que os dados inseridos não sejam confundidos com instruções executáveis.

Exemplo:

```
ORG 0010h
DB 05h, 0CFh, "ISTO E UM TESTE", 00H ; esta diretiva insere diretamente a partir da
                                     ; posição de memória 0010h os seguintes
                                     ; códigos hexadecimais ( 05, CF, 49, 53, 54,
                                     ; 4F, 20, 45, 20, 55 ,4D, 20, 54, 45, 53, 54, 45,
                                     ; 00)
```

3) Diretiva DW - Define Word

DW dataword1 ***[, dataword2, [dataword3...]]***
DW "string1" ***[. string2 [, string3...]]***

A diretiva DW permite ao programador inserir palavras de dados (2 bytes) diretamente no programa na posição de memória corrente.
Os valores numéricos de 16 Bits (2 Bytes) são inseridos respeitando-se o seu formato (decimal, hexadecimal, binário, octal). Se mais de um valor for inserido eles devem vir separados por vírgula. Se apenas um Byte for inserido o MSB será adotado como 00.
Caracteres ASCII isolados ou “Strings” de caracteres ASCII devem estar contidos entre aspas. Se apenas um caractere ASCII for inserido, o LSB será 00.

Exemplo:

```
ORG 0100h
DW 567Fh, "TESTE", 05H, “A”     ; esta diretiva insere diretamente a partir da
                                ; posição de memória 0100h os seguintes
                                ; códigos hexadecimais (56, 7F, 54, 45, 53, 54, 45, 00,
                                ; 05, 41, 00)
```

Obs: Esta diretiva deve ser colocada sempre depois do fim lógico do programa para que os dados inseridos não sejam confundidos com instruções executáveis.

4) Diretiva EQU (=) – (Equate) Igual

Variable **EQU** ***value***
Variable = value

Atribui um valor (value) à uma Variável (Variable). A diretiva EQU e o sinal = são sinônimos e podem ser usadas para atribuir um valor específico à Variável. A Variável só pode receber um único valor a menos que seja declarada como PUBLIC. O valor pode ser um valor numérico ou uma expressão. Uma vez declarado o valor da variável este não poderá mudar.

Exemplo:

```
        ORG 0
Controle EQU 10h ; atribui 10h à variável Controle
Controle2 = 20h ; atribui 20h à variável Controle2
        MOV A, #Controle
```

Obs: Deve ser declarada no programa anteriormente ao uso da Variável. Uma dica é sempre colocar esta diretiva no início do programa, antes da primeira instrução executável.

Exemplo de escrita de um Programa Fonte:

```
;*********************************************************************
; Título do Programa: Programa Principal                              *
; Este é um programa exemplo para mostrar como escrever um código de programa  *
; fonte e comentá-lo, facilitando  futuras correções.                 *
;*********************************************************************

; Atribuição das variáveis do programa

Var1    EQU 30h             ; Esta variável estabelece o início da contagem
Var2    EQU 02h             ; Variável que determina o número de incrementos
;*********************************************************************

        ORG 0               ; Início do programa principal no endereço 0000h

Init:   MOV A, #Var1        ; Usar o Valor de Var1 para iniciar a contagem
        ADD A, #Var2        ; Incrementar a contagem de 02 unidades
Loop:   ACALL Rot1          ; Chamar a Sub-rotina de análise dos dados
        SJMP Loop           ; Voltar para o Loop e permanecer calculando

; Fim Lógico do Programa Principal. (O Fim Lógico não permite que o programa
; principal ultrapasse este ponto, evitando executar lixo que esteja residente na
; memória)
;*********************************************************************


;*********************************************************************
; Área das Sub-rotinas: as sub-rotinas devem ficar após o Fim Lógico do programa *
; Principal, pois, serão chamadas por instruções específicas quando for necessário  *
; executá-las.                                                        *
;*********************************************************************
```

```
;*********************************************************************
; Sub-rotina Rot1 :                                                  *
; Esta sub-rotina analisa os dados e executa o cálculo dos máximos valores  *
;*********************************************************************

Rot1:   ADD A, #Var1
        MOV R0,#Var2
        DJNZ R0, Pulo              ; Loop de controle do sistema
        SJMP Rot1
Pulo:   RET

; Fim da Sub-rotina Rot1.
;*********************************************************************


;*********************************************************************
; Área de Dados para criação da Tabela                               *
;*********************************************************************

Tab1:   DB 12h, 45h, 0DFh, “abcd”
Tab2:   DW “Erro. Entrar com outro Valor”, 34h, 5656h

; Fim da área de dados
;*********************************************************************

;*********************************************************************
; Fim físico do Programa. Define para o programador e para o compilador a região *
; de código de todo o programa.                                      *
; Atenção: O Fim Físico é apenas simbólico. Não é um código que faz o programa  *
; parar! É preciso ter um fim Lógico!                                *
;*********************************************************************
END
```