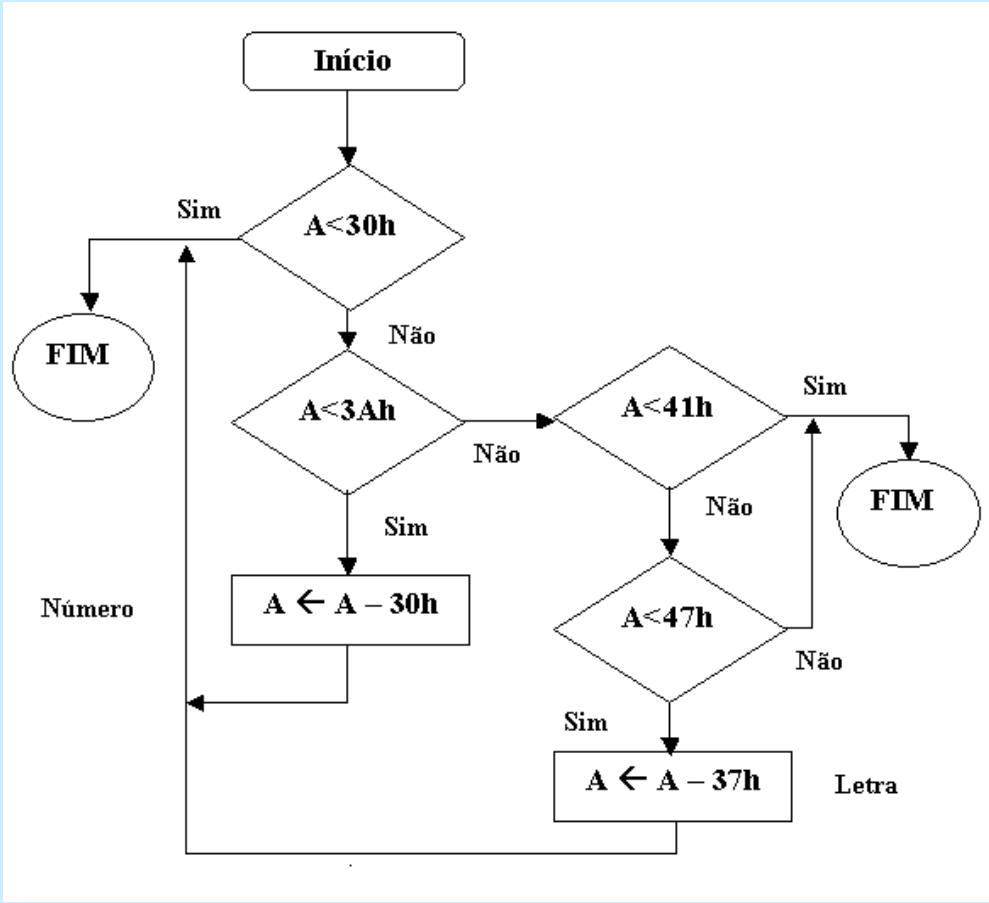
SEL-433 Aplicação de Microprocessadores I

Programação de Microprocessadores



Programação de Microprocessadores

- Microprocessadores são **‘Máquinas de Estado Seqüenciais Síncronas’** que operam mediante a execução de uma seqüência de códigos binários armazenados em memória.

• As ordens ou comandos compreendidos por um determinado Microprocessador, são **INSTRUÇÕES** seqüencialmente armazenadas na Memória.

• Ao conjunto de Instruções compreendidos por um determinado Microprocessador dá-se o nome de "**INSTRUCTION SET**".

• Cada Microprocessador tem seu próprio Instruction Set que é em geral, diferente do Instruction Set de outro Microprocessador de fabricantes diferentes.

• Uma seqüência de Instruções do Instruction Set, armazenadas na memória e que realiza alguma operação, recebe o nome de **PROGRAMA**.

• Cada Instrução do Microprocessador é um código binário formada em geral por um ou mais Bytes.

• A cada código binário equivalente a uma Instrução está associado um **Mnemônico** para facilitar a compreensão da função que a Instrução executa.

• Ao conjunto de Instruções e seus Mnemônicos equivalentes dá-se o nome de **LINGUAGEM ASSEMBLY**.

Fluxograma

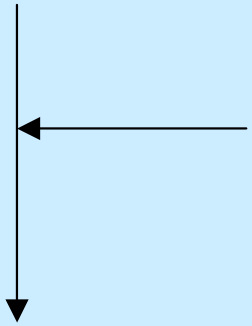
• Para a documentação lógica de um Programa em Assembly utiliza-se um **Fluxograma ou Diagrama de Blocos.**

• Cada bloco do Fluxograma equivale a um sub-conjunto do Instruction Set do Microprocessador.

• O Fluxograma é uma forma de se implementar logicamente um programa, antes que o mesmo seja codificado na Linguagem Assembly do Microprocessador.

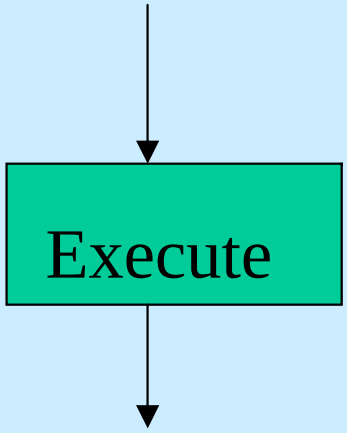
Fluxograma

- Linhas de Fluxo do Programa



- Mostram a seqüência de execução das Instruções.
- Cada Bloco do Fluxograma possui apenas uma linha de Fluxo de Entrada e uma ou duas de saída

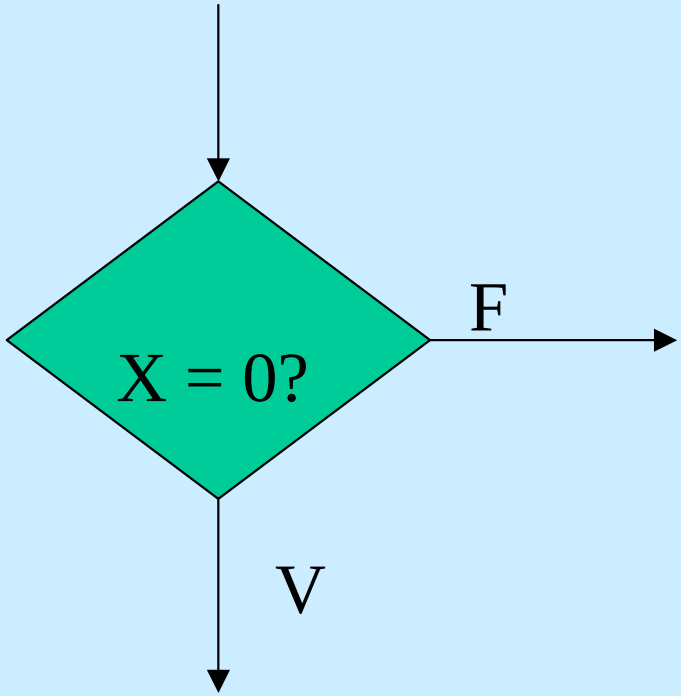
- Bloco de Processo



- Equivalem às Instruções que realizam alguma operação do tipo:
  - Movimento de Dados
  - Operação Aritmética
  - Operação Lógica

Fluxograma

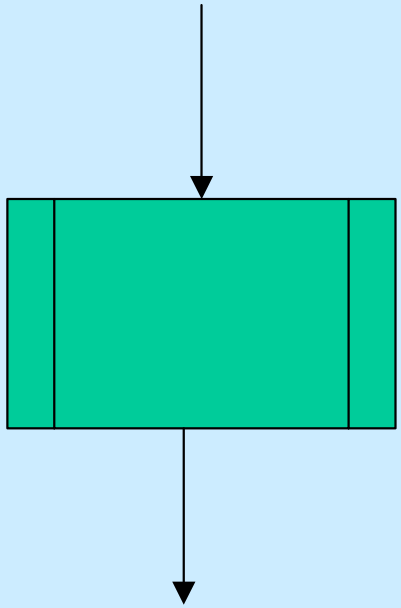
- Bloco de Decisão



- Equivale às Instruções que decidem sobre o Fluxo do Programa.
- Se a função dentro do bloco for Verdadeira(V) o programa continua abaixo, se for Falsa(F) o programa muda o fluxo.

Fluxograma

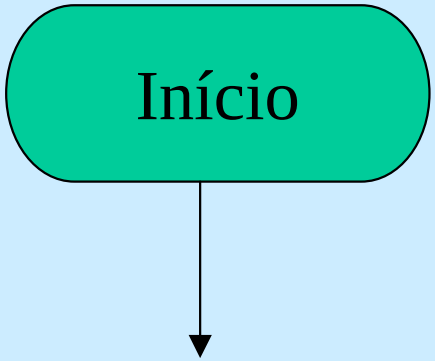
- Processo Pré-definido



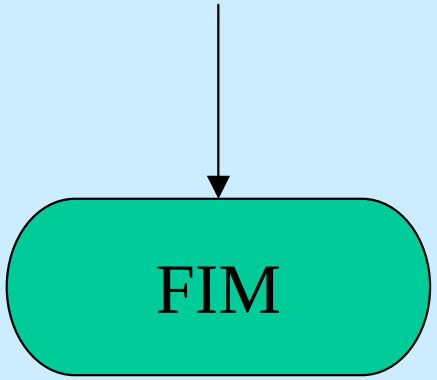
- Equivale às Instruções que mandam executar uma Sub-rotina armazenada em outro lugar da Memória.
- Observe que quando a sub-rotina termina, o fluxo do programa continua normalmente.

Fluxograma

- Bloco de Início de Programa



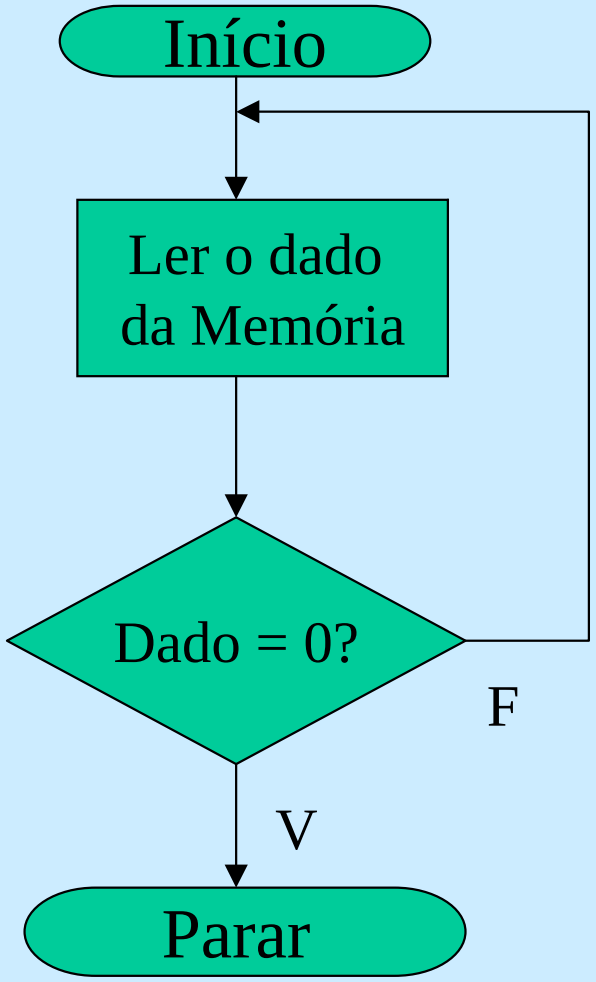
- Bloco de Fim de Programa



- O Bloco de Início de Programa não equivale a uma Instrução específica do Instruction SET.

- O Bloco de FIM equivale a uma instrução que termina o Programa. É chamado de **FIM LÓGICO** do Programa.

Exemplo de Fluxograma de um Programa de Microprocessador



• O programa ao lado deve Ler um Dado da memória, verificar se é igual a zero. Se não for zero, continua em LOOP. Se for zero para o programa.

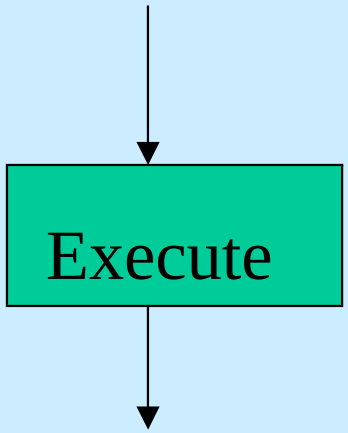
Codificação Assembly

• Para Codificar um Programa escrito através de um Fluxograma, deve-se escolher o Microprocessador, ou seja, conhecer seu Conjunto de Instruções.

• Os Microcontroladores da família MCS-51 serão os dispositivos a serem aplicados nesta disciplina.

Codificação Assembly do 8051

• Instruções equivalentes
ao Bloco de Processo

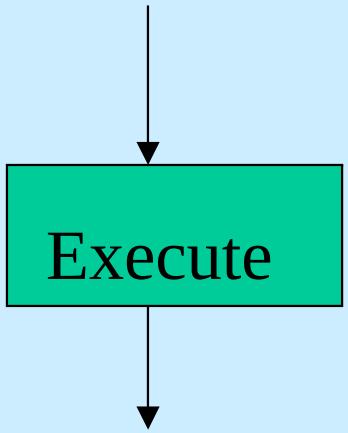


• Instruções Aritméticas

SUBB A, direct	**ADD A, Rn**
INC A	**DEC A**
DA A	

Codificação Assembly do 8051

• Instruções equivalentes
ao Bloco de Processo

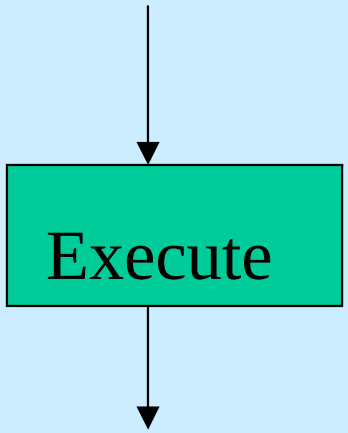


• Instruções Lógicas

ANL A, Rn	**ORL A, direct**
XRL A, #data	
CLR A	**CPL A**
RL A	**SWAP A**

Codificação Assembly do 8051

- Instruções equivalentes ao Bloco de Processo



- Instruções de Transferência de Dados

MOV A, Rn

MOVC A, @A+DPTR

MOVX A,@DPTR

PUSH direct

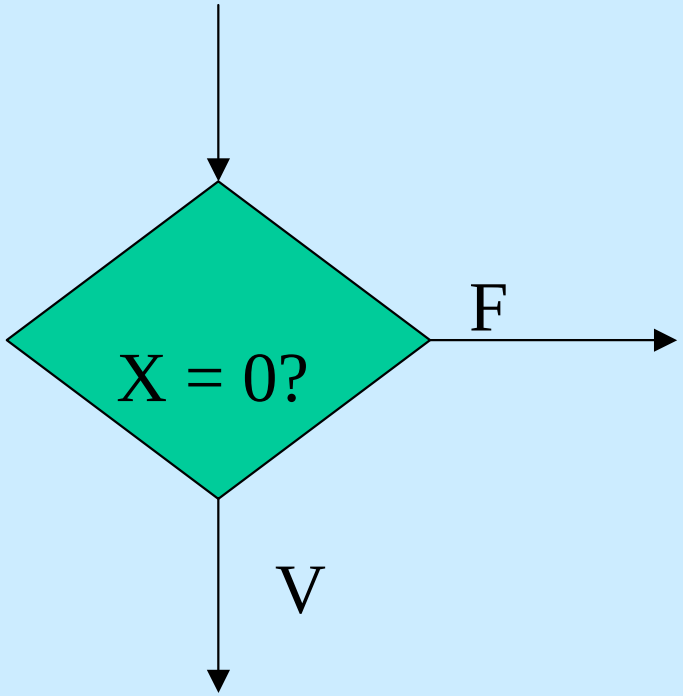
POP direct

XCH A, Rn

Codificação Assembly do 8051

- Instruções equivalentes ao Bloco de Decisão

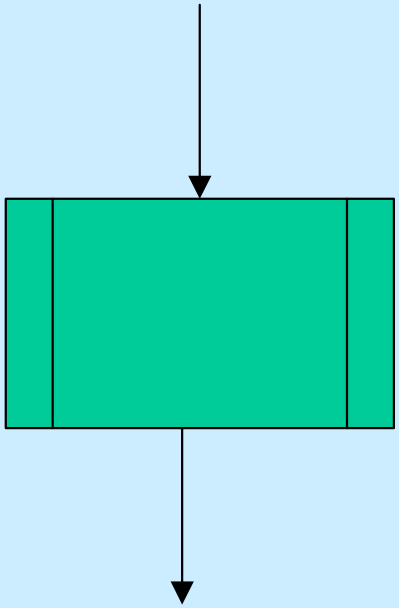
- Instruções de Desvio



JZ	**rel**
JNZ	**rel**
CJNE	**A, direct, rel**
JC	**rel**
JNC	**rel**
JB	**bit, rel**
JNB	**bit, rel**
DJNZ	**Rn, rel**

Codificação Assembly do 8051

• Instruções equivalentes ao Bloco de Processo Pré-definido



• Instruções de Sub-Rotina

LCALL addr16

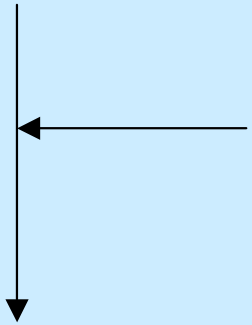
ACALL addr11

RET

RETI

Codificação Assembly do 8051

- Instruções equivalentes a Mudança de Fluxo



- Instruções de Saltos

LJMP	**addr16**
AJMP	**addr11**
SJMP	**rel**
JMP	**@A+DPTR**

Modos de Endereçamento do 8051

1. Endereçamento Imediato

   • Opera sobre o dado localizado na própria instrução

• Identificado através do sinal #

• Exemplo: **ADD A,#30**

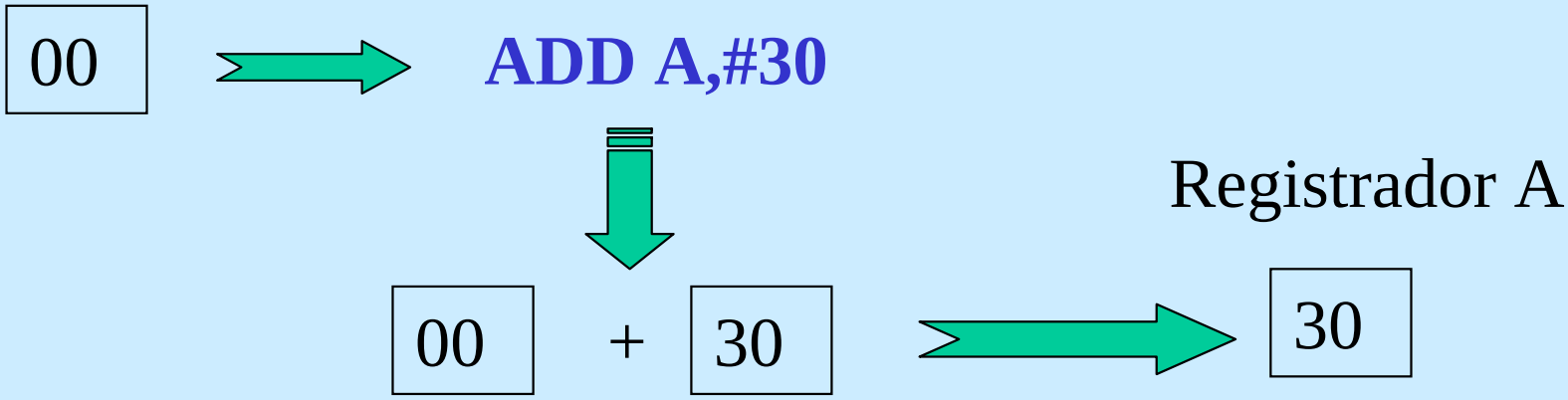
O dado 30 é somado ao Registrador A

Modos de Endereçamento do 8051

1. Endereçamento Imediato

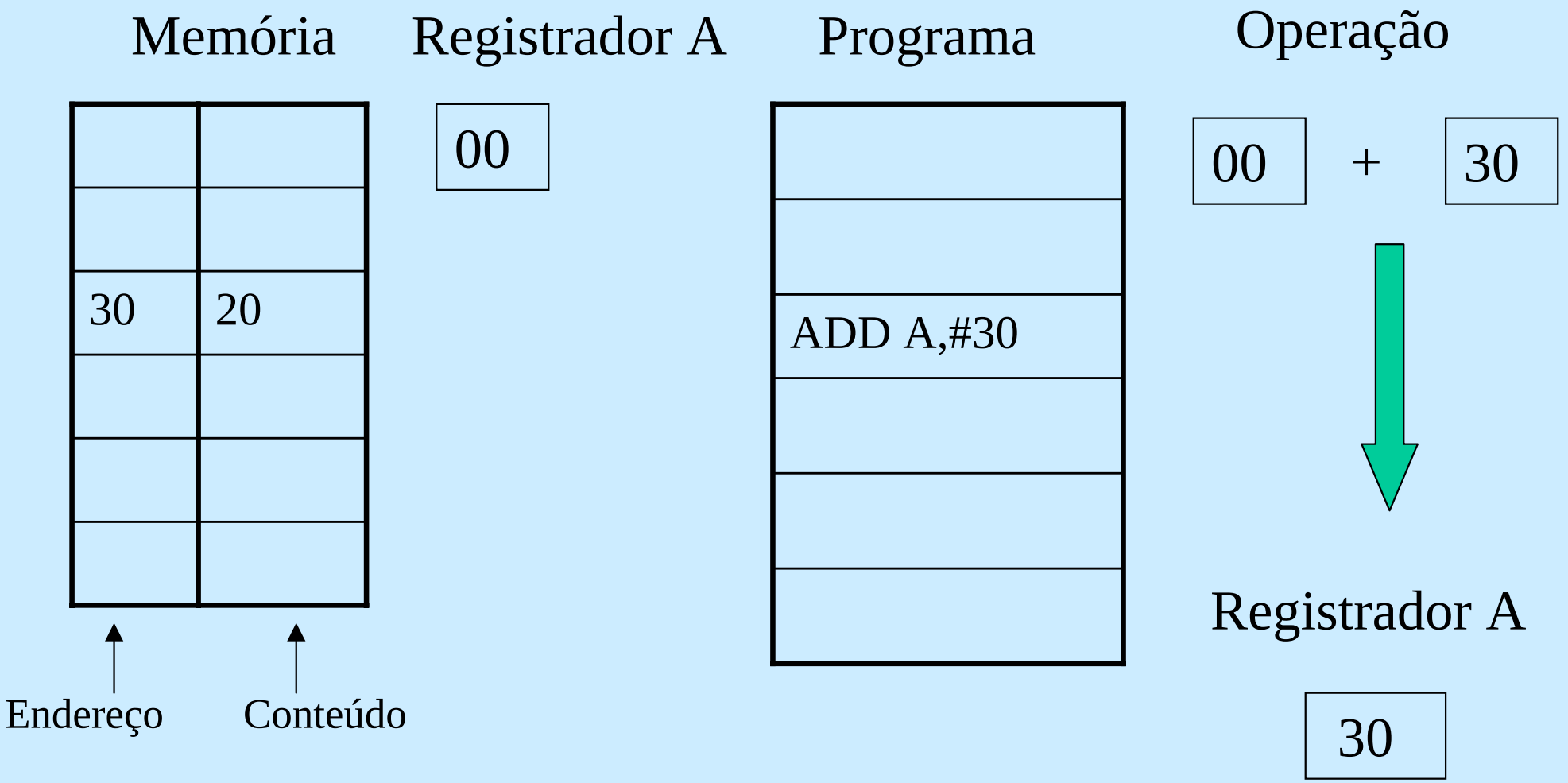
ADD A,#30

Registrador A



Modos de Endereçamento do 8051

ADD A,#30



Modos de Endereçamento do 8051

2. Endereçamento Direto

- Opera sobre o dado cujo endereço está na instrução

- Exemplo:

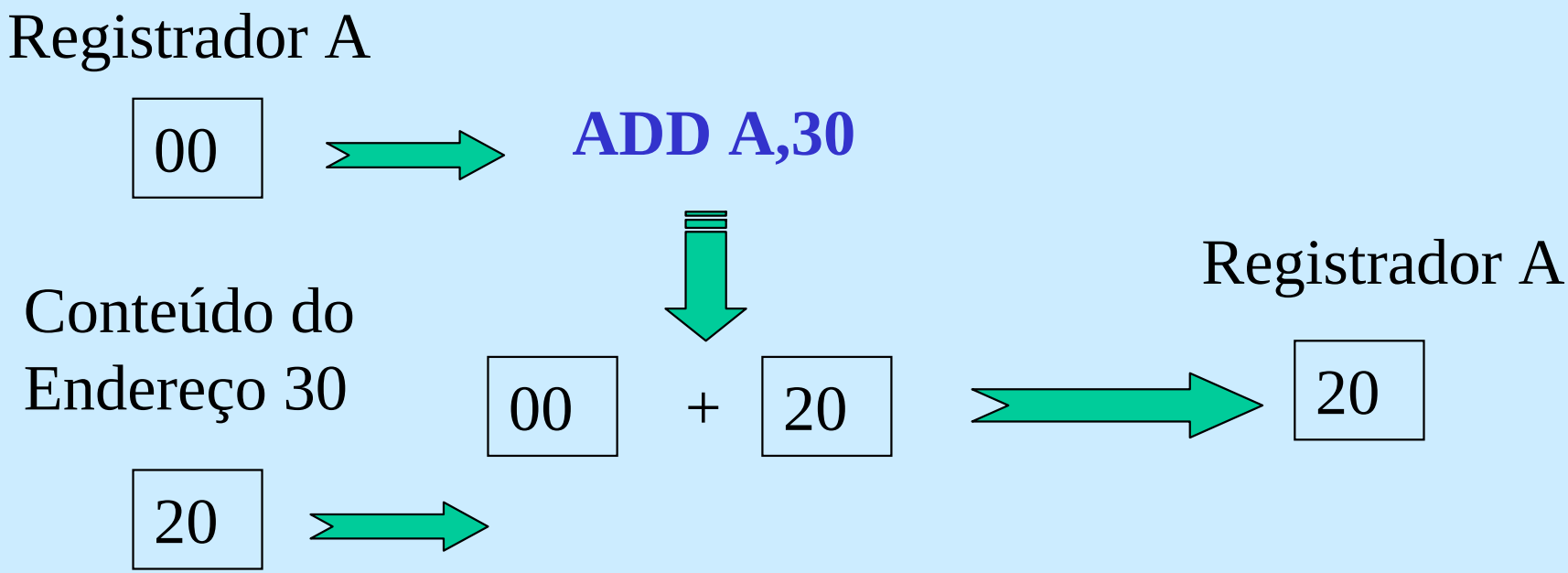
ADD A,30

O dado armazenado no endereço 30 é somado ao Registrador A

Modos de Endereçamento do 8051

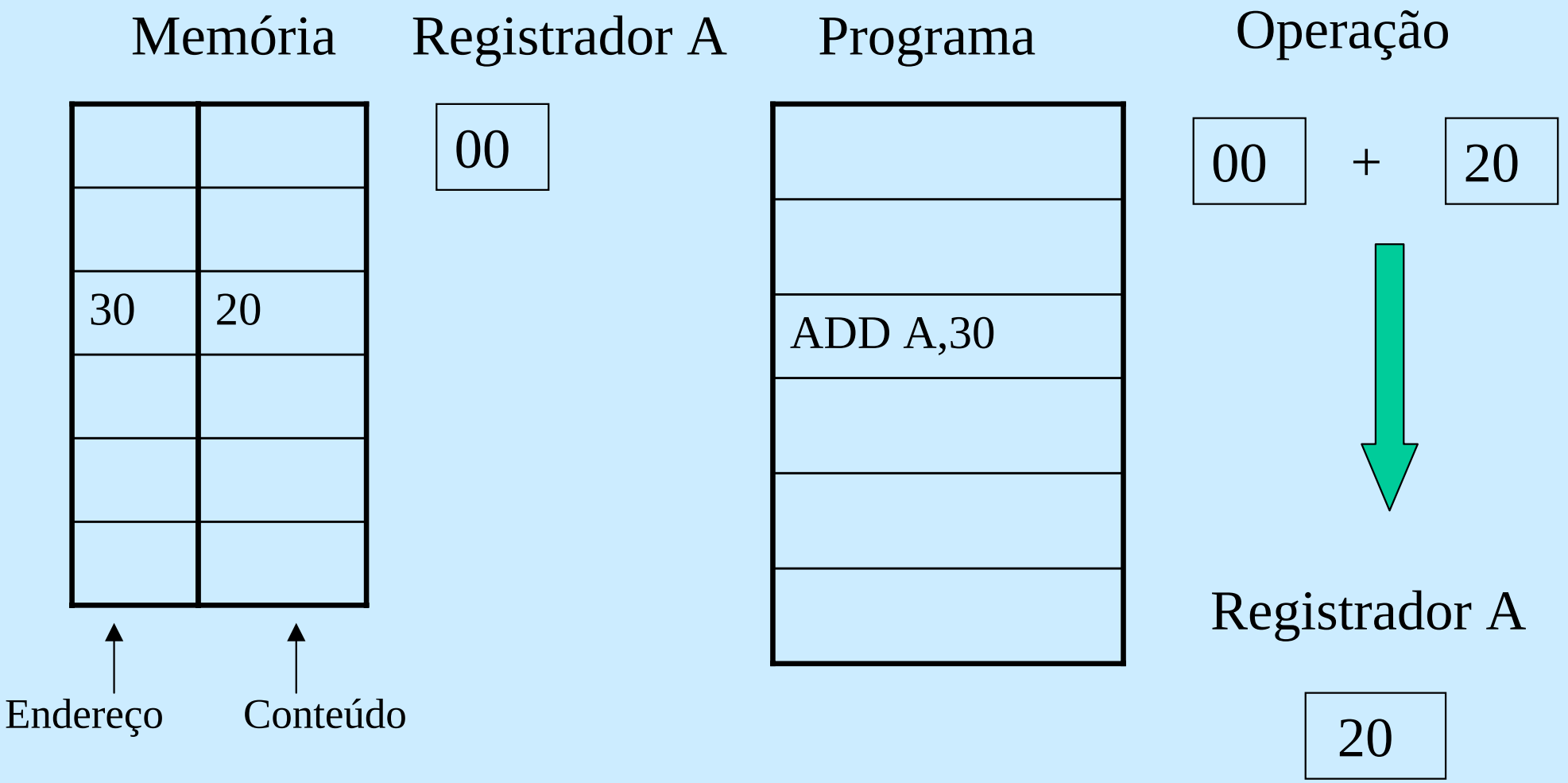
2. Endereçamento Direto

ADD A,30



Modos de Endereçamento do 8051

ADD A,30



Modos de Endereçamento do 8051

2. Endereçamento Indireto

• Opera sobre o dado cujo endereço está armazenado em um Registrador apontado na instrução

• Identificado através do sinal @

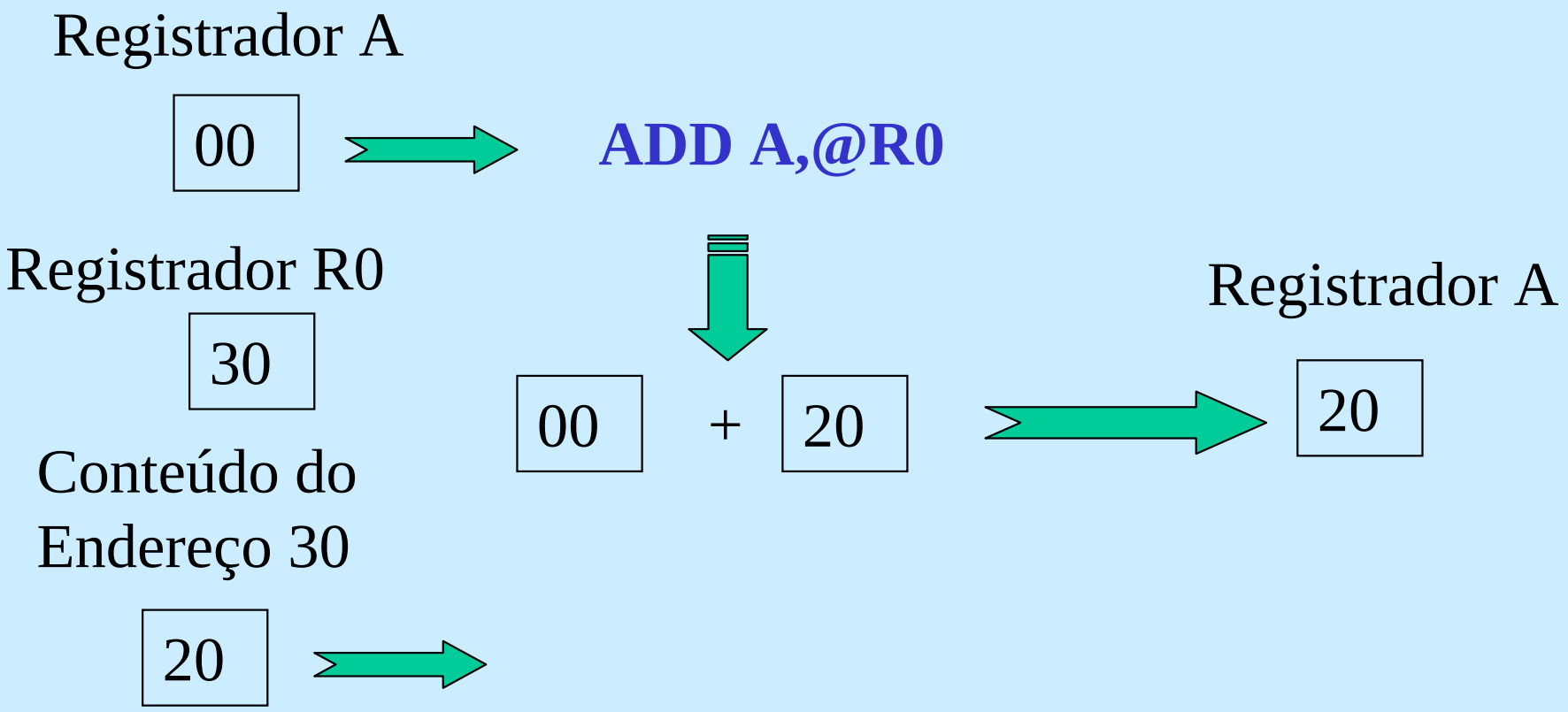
• Exemplo: **ADD A,@R0**

O dado armazenado no endereço apontado pelo Registrador R0 é somado ao Registrador A

Modos de Endereçamento do 8051

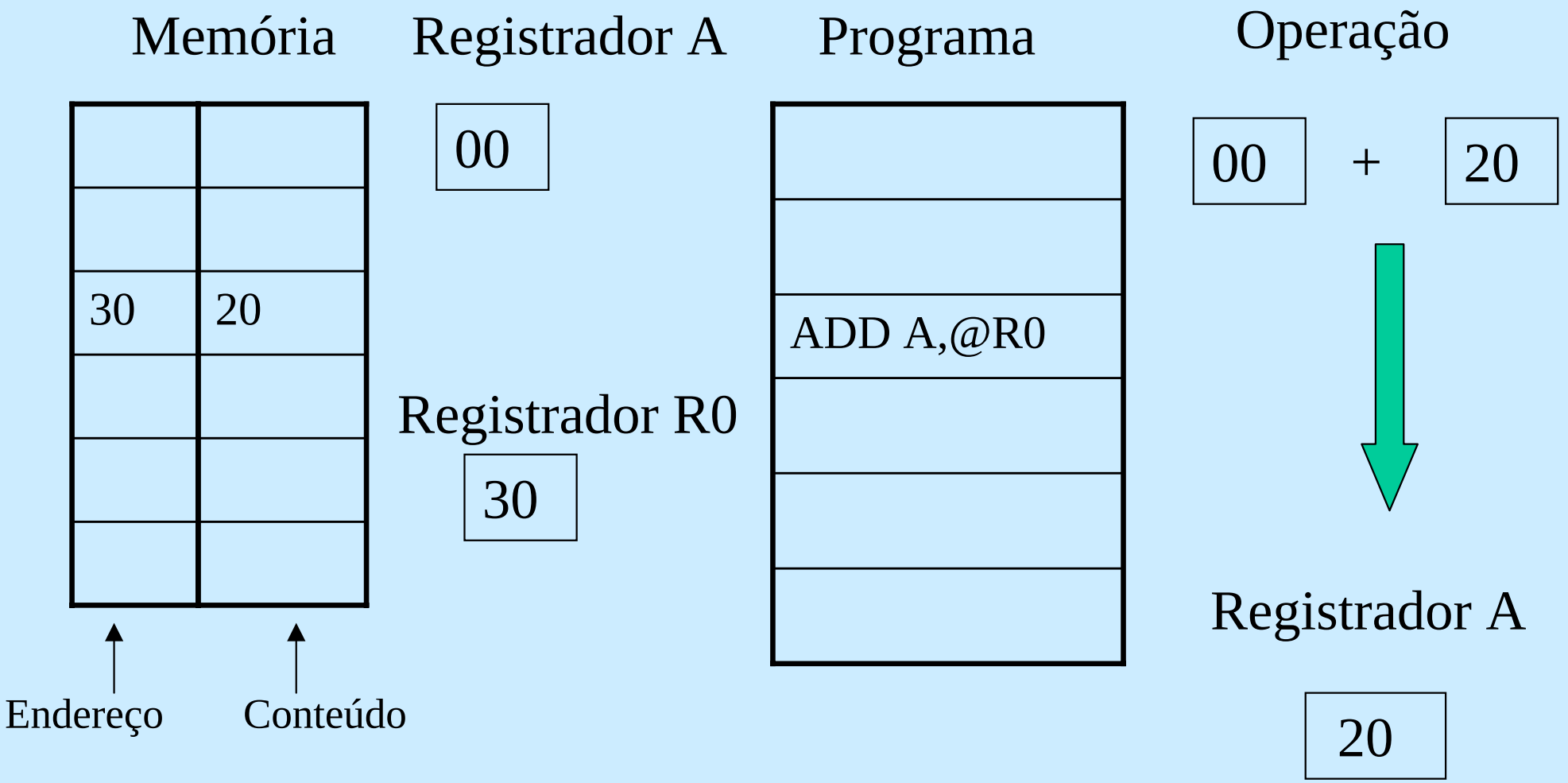
2. Endereçamento Indireto

ADD A,@R0

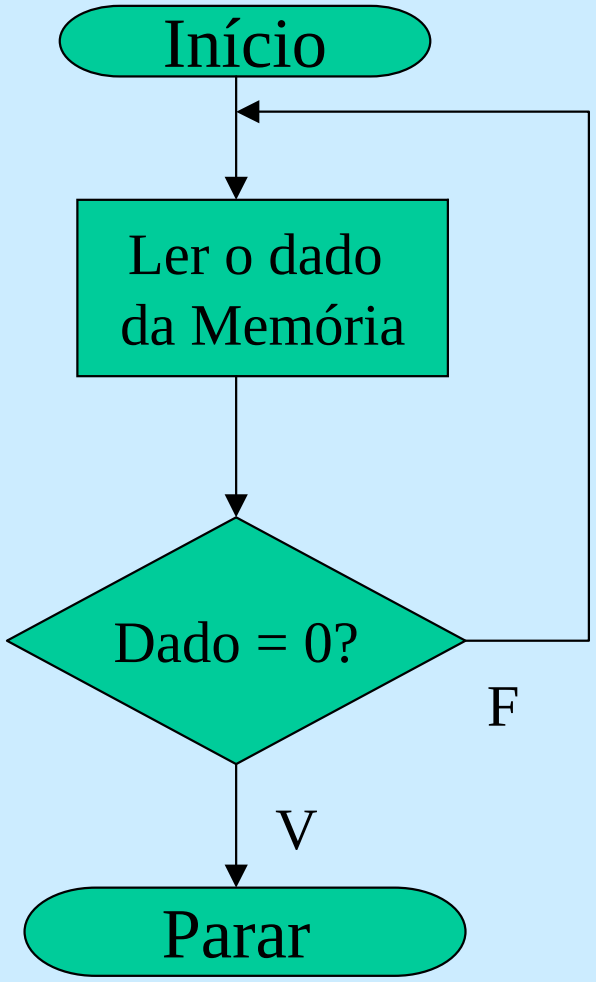


Modos de Endereçamento do 8051

ADD A,@R0



Exemplo de um Programa Assembly do 8051

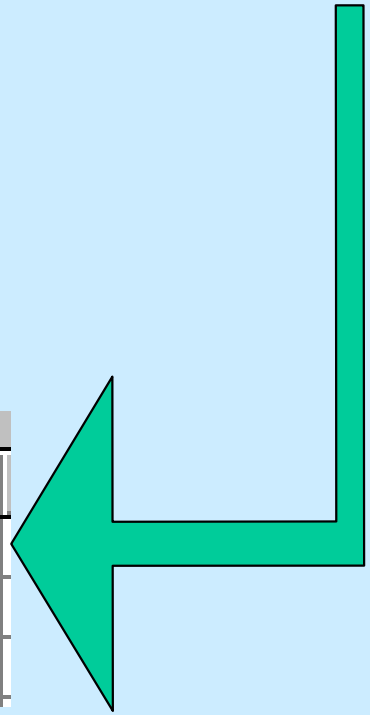



```
                ORG 0
LOOP:
                MOV A,30H


                CJNE A,#00,LOOP


AQUI:           SJMP AQUI
```

Exemplo de um Programa Assembly do 8051

Mnemônicos (Programa Assembly)

```
        ORG 0
LOOP:   MOV A,30H
        CJNE A,#00,LOOP
AQUI:   SJMP AQUI
```



Código Compilado (Opcode)

Addr	Opcodes	ASC	Label	Disassembly
0000	E5 30	å0	LOOP	MOV A,30h
0002	B4 00 FB	´û		CJNE A,#00h,LOOP
0005	80 FE	€þ	AQUI	SJMP AQUI

Exemplo de um Programa Assembly do 8051

Memória de Programa

00	E5
01	30
02	B4
03	00
04	FB
05	80
06	FE

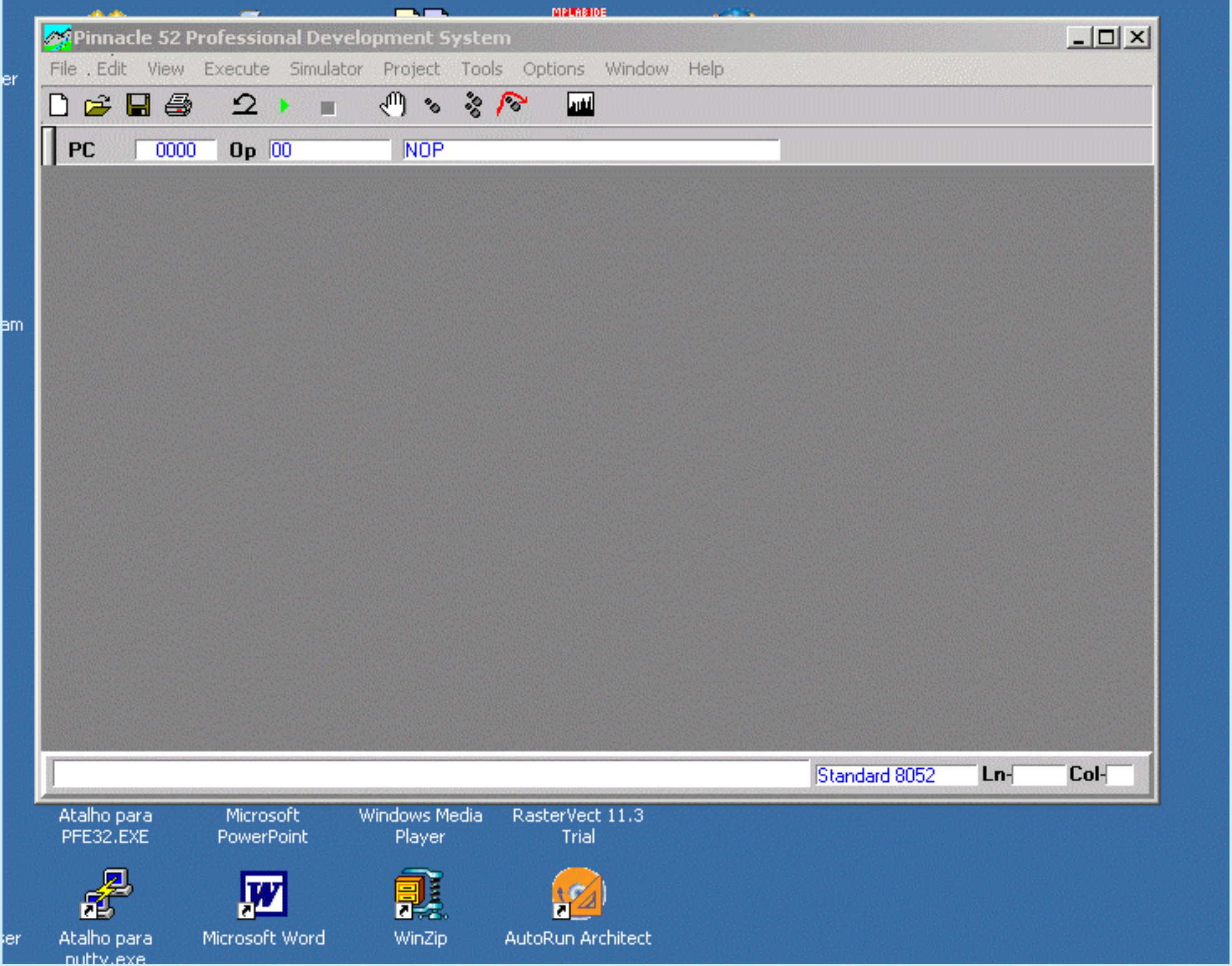
Endereço Conteúdo

Addr	Opcodes	ASC	Label	Disassembly
0000	E5 30	å0	LOOP	MOV A,30h
0002	B4 00 FB	´ûû		CJNE A,#00h,LOOP
0005	80 FE	€þ	AQUI	SJMP AQUI

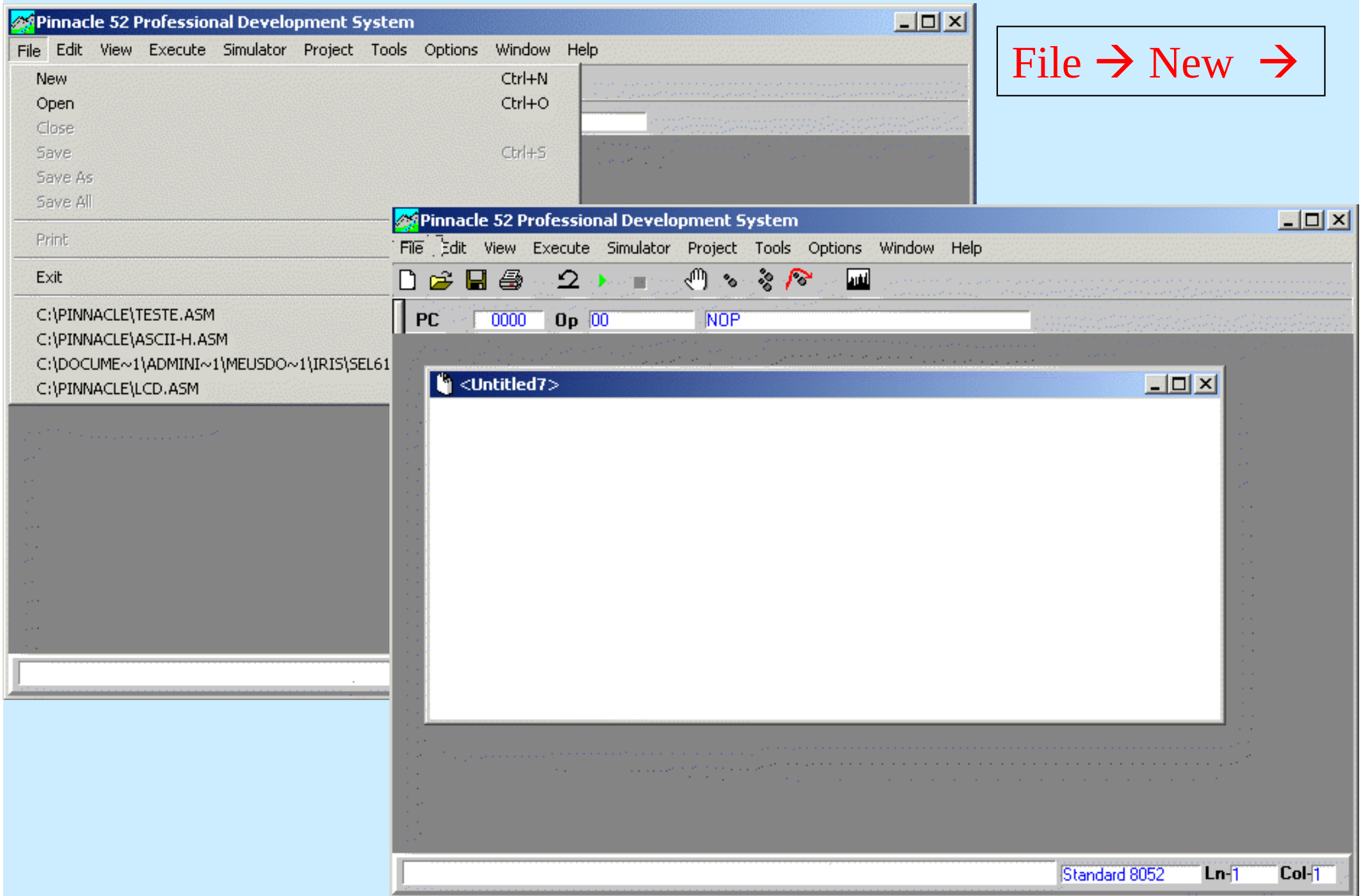
Ambiente de desenvolvimento de Programação Assembly

PINNACLE

http://www.vaultbbs.com/pinnacle/

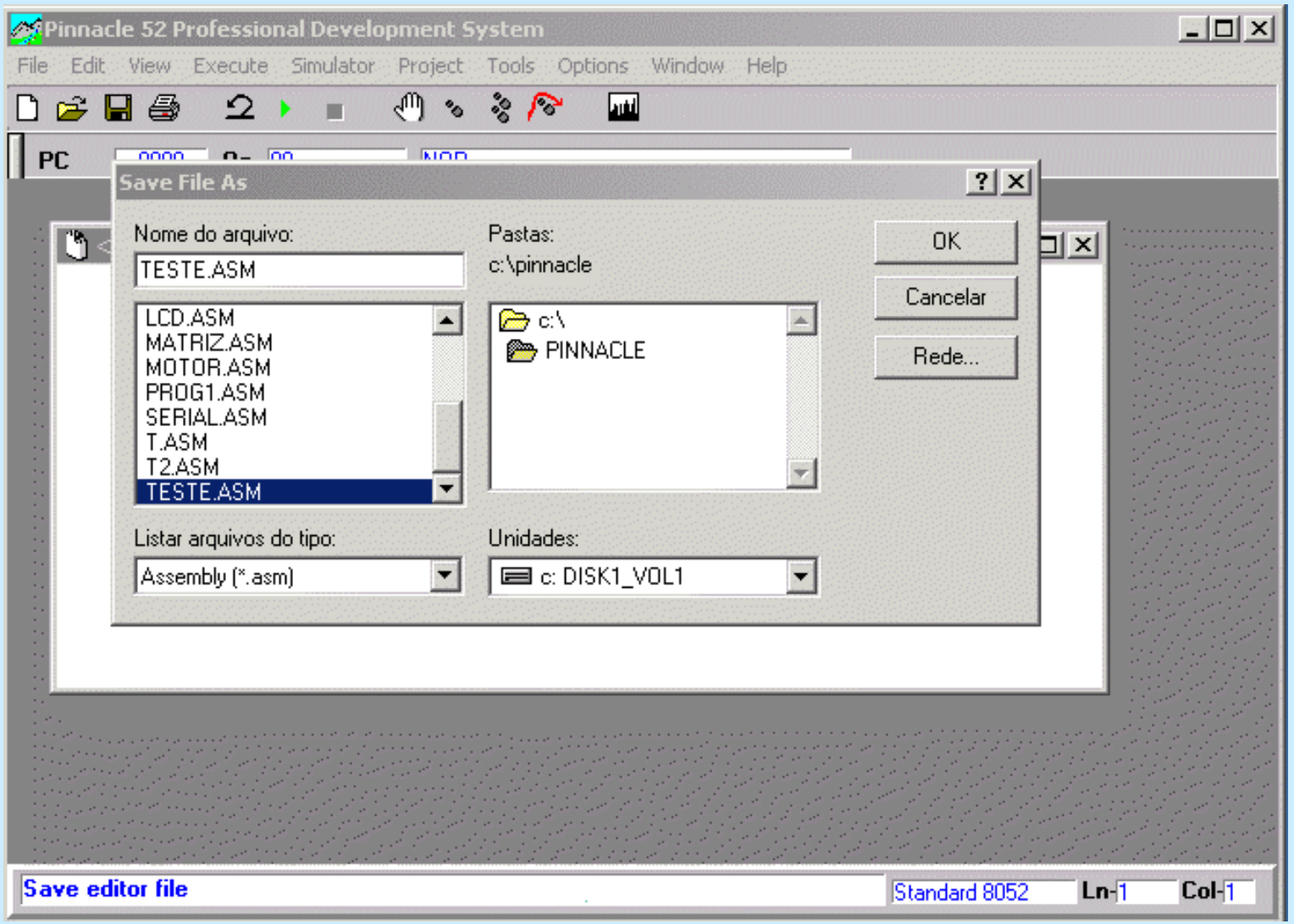

Pinnacle 52 Professional Development System
File . Edit View Execute Simulator Project Tools Options Window Help
PC 0000 Op 00 NOP
Standard 8052 Ln- Col-
Atalho para PFE32.EXE
Microsoft PowerPoint
Windows Media Player
RasterVect 11.3 Trial
Atalho para putty.exe
Microsoft Word
WinZip
AutoRun Architect

Para escrever um Programa em Assembly



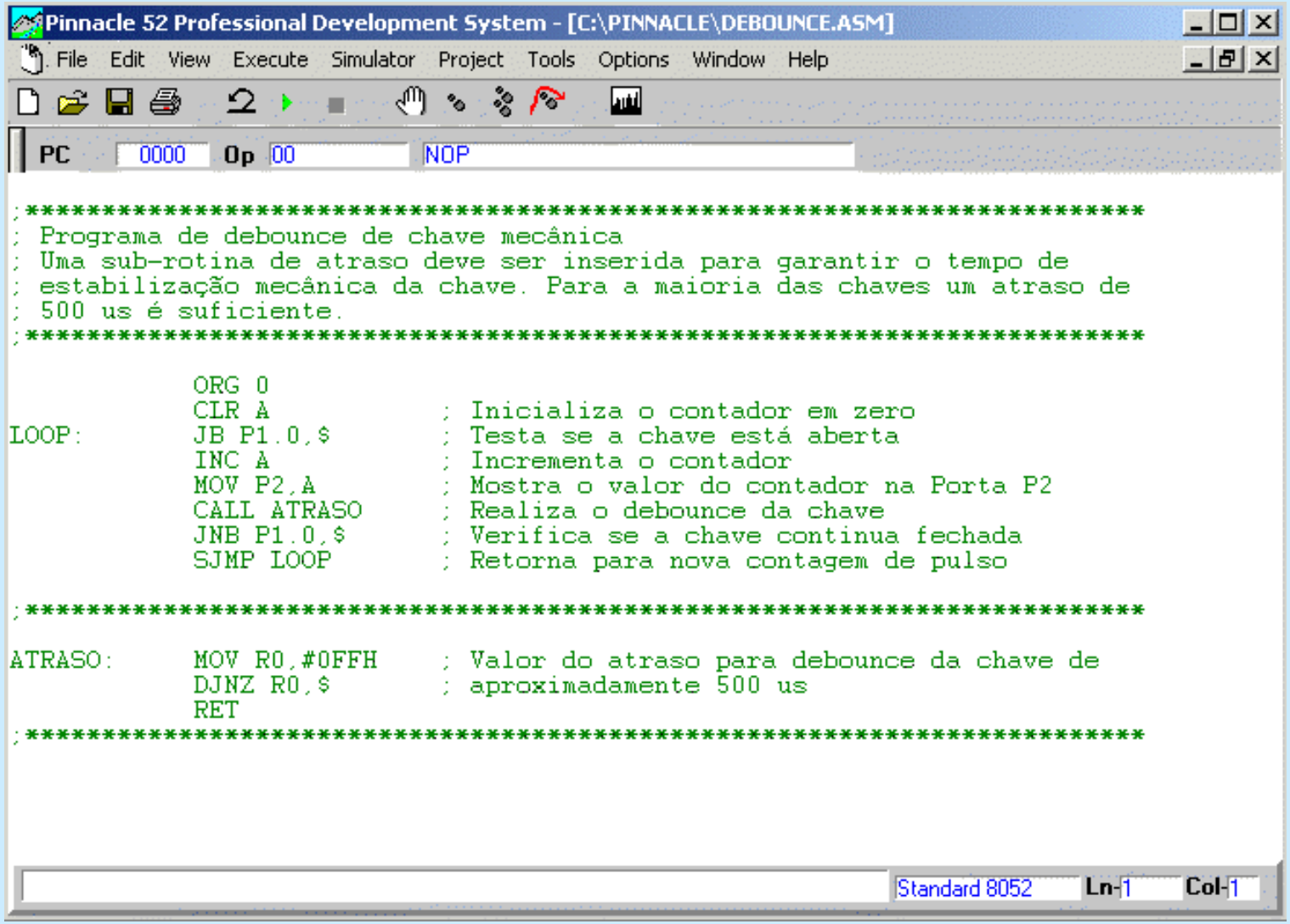
File → New →

File → Save →



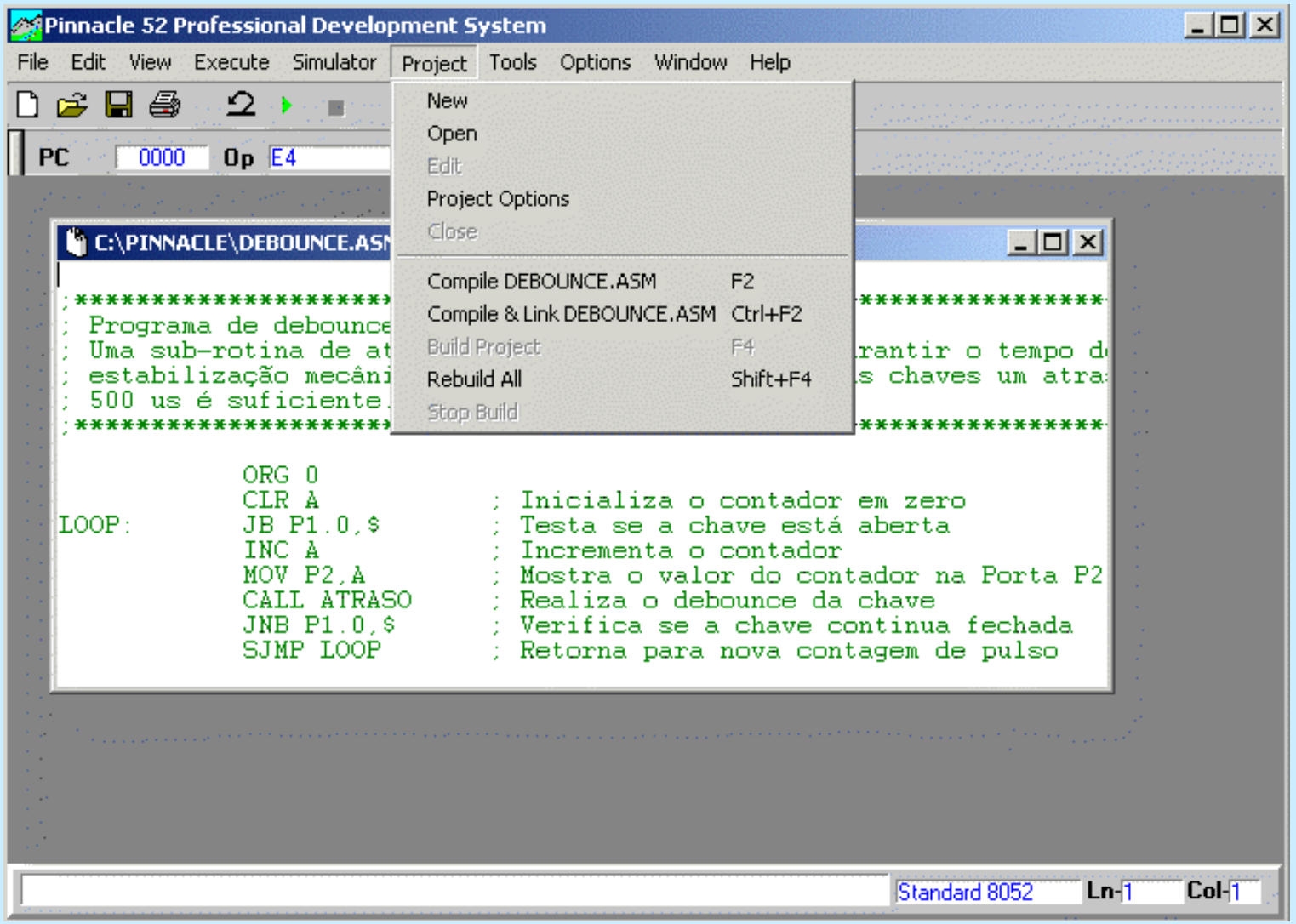
Obs: O arquivo deve ser salvo como “Nome.asm” em um diretório próximo ao raiz. O Nome deve ser curto pois o Pinnacle não lida com nomes grandes de arquivos.

Exemplo de um Programa em Assembly: (Código Fonte)

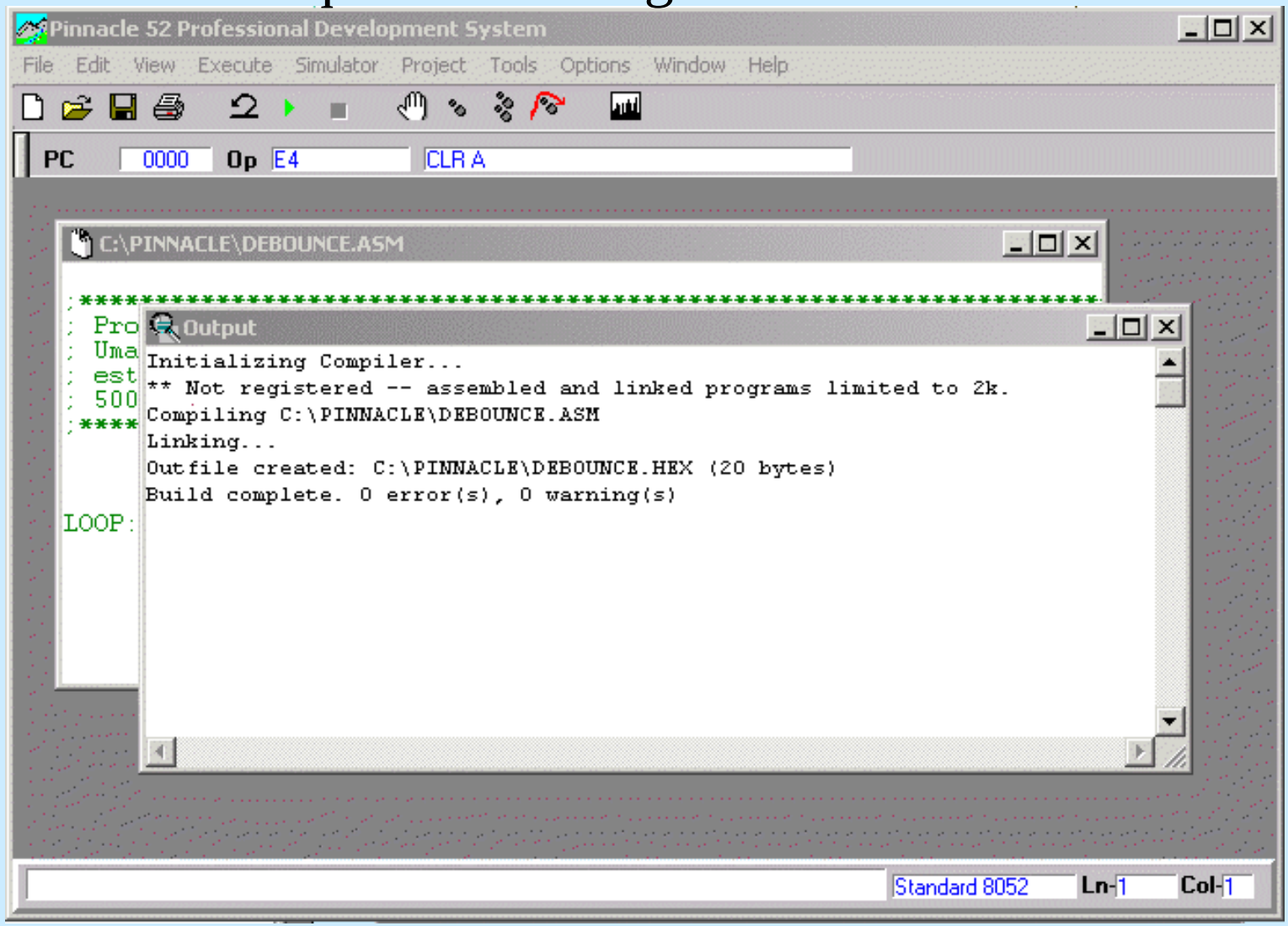


Para Compilar o Código Fonte e gerar o Código Objeto:

Project → Compile & Link →

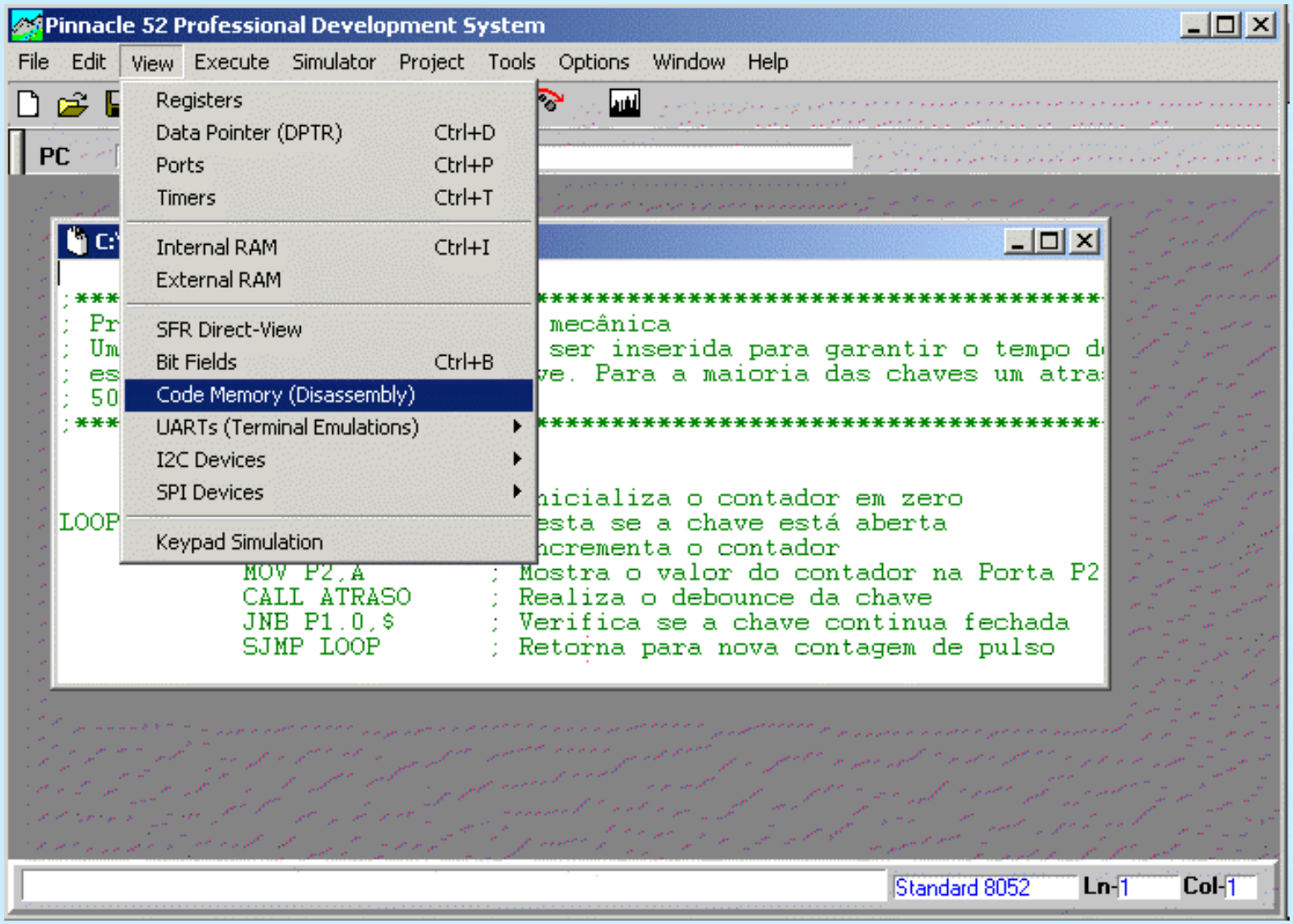


O Código Objeto é gerado no formato .HEX , no mesmo diretório do arquivo do Código Fonte.



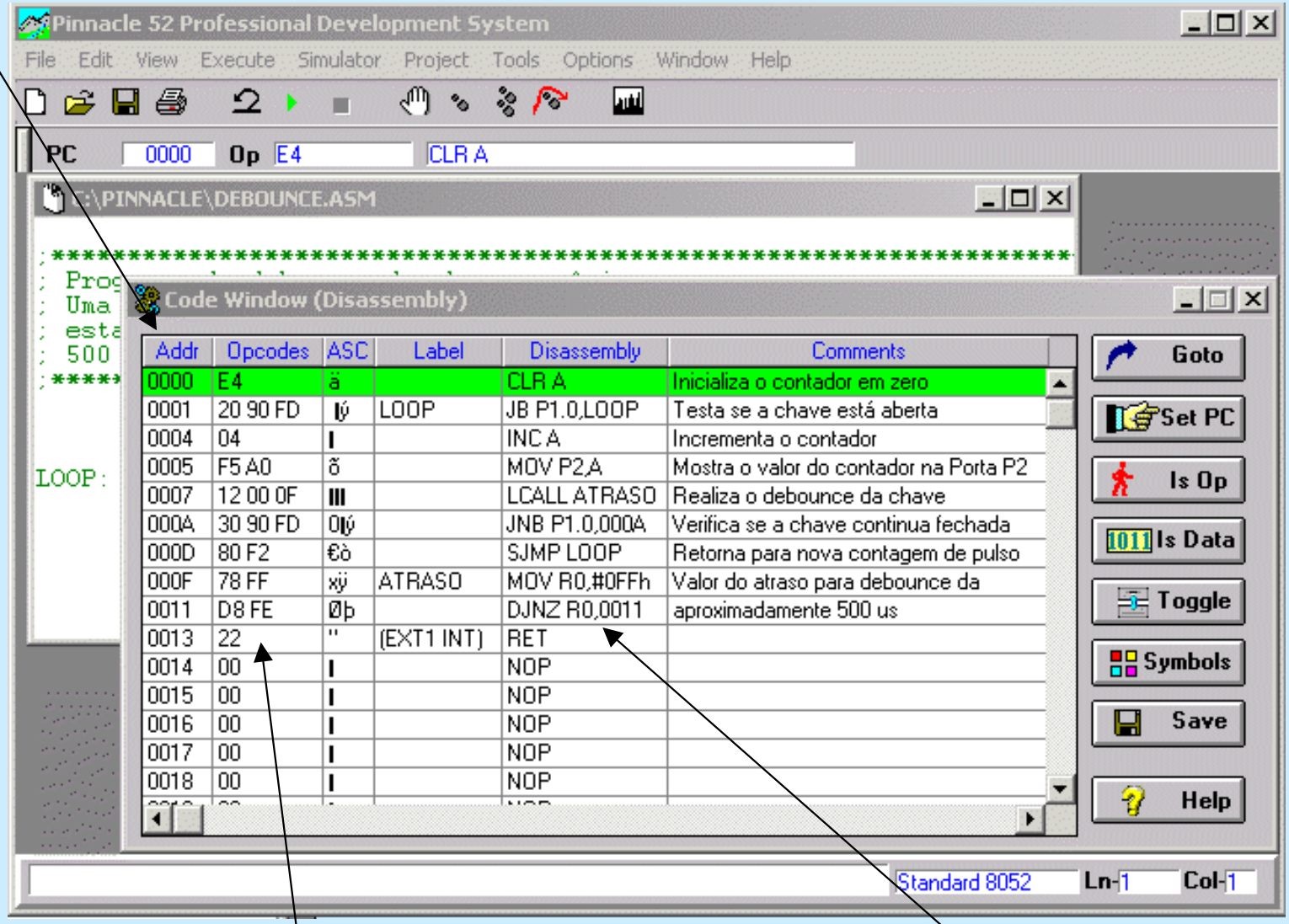
A janela “Output” mostra se a compilação não teve erros ou em que linhas do Código Fonte existem erros.

Para visualizar o Código Objeto
(Programa Compilado):



View → Code Memory (Disassembly) →

Endereço da Memória de Programa (Hexadecimal)



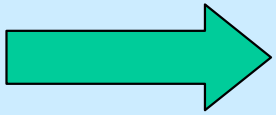
Conteúdo da Memória de Programa (Instruções Compiladas)

Mnemônicos das Instruções em Assembly

Re-escrevendo o Programa Objeto (Compilado):

Endereço da Memória de Programa (16 Bits)

Addr	Opcodes
0000	E4
0001	20 90 FD
0004	04
0005	F5 A0
0007	12 00 0F
000A	30 90 FD
000D	80 F2
000F	78 FF
0011	D8 FE
0013	22
0014	00
0015	00
0016	00
0017	00



0000	E4
0001	20
0002	90
0003	FD
0004	04
0005	F5
0006	A0
0007	12
0008	00
0009	0F
000A	30
000B	90
000C	FD
000D	80
000E	F2
000F	78
0010	FF
0011	D8
0012	FE
0013	22
0014	00

Conteúdo da Memória de Programa (8 Bits) → Instruções Binárias

Programa Fonte (Assembly) Formato Texto → Compilador e Linker → Programa Objeto (Código Compilado) Formato Binário

```
            ORG 0
            CLR A
LOOP:       JB P1.0,$
            INC A
            MOV P2,A
            CALL ATRASO
            JNB P1.0,$
            SJMP LOOP

;*****************************

ATRASO:     MOV R0,#0FFH
            DJNZ R0,$
            RET
;*****************************
```

ORG 0 → Origem do Programa Objeto na Posição 0000 da Memória de Programa

Addr	Opcodes
0000	E4
0001	20 90 FD
0004	04
0005	F5 A0
0007	12 00 0F
000A	30 90 FD
000D	80 F2
000F	78 FF
0011	D8 FE
0013	22
0014	00
0015	00
0016	00
0017	00

A primeira Instrução do Programa (CLR A) cujo código binário é (E4) será armazenada na Posição 0000 da Memória de Programa.

??? Qual o código binário da segunda Instrução do Programa e onde ele está armezenado???

??? Qual o código binário da segunda Instrução do Programa e onde ele está armezenado???

```
                ORG 0
                CLR A
LOOP:           JB P1.0,$
                INC A
                MOV P2,A
                CALL ATRASO
                JNB P1.0,$
                SJMP LOOP

;*****************************

ATRASO:         MOV R0,#0FFH
                DJNZ R0,$
                RET
;*****************************
```

A segunda Instrução é: **JB P1.0,$**

Que será armazenada a partir da segunda Posição da Memória de Programa (Endereço 0001)

Addr	Opcodes
0000	E4
0001	20 90 FD
0004	04
0005	F5 A0
0007	12 00 0F
000A	30 90 FD
000D	80 F2
000F	78 FF
0011	D8 FE
0013	22
0014	00
0015	00
0016	00
0017	00

Como esta Instrução é codificada usando 3 Bytes (20 90 FD), e cada posição de memória armazena apenas um Byte, ela será armazenada nas Posições 0001 0002 0003

0000	E4
0001	20
0002	90
0003	FD
0004	04
0005	F5
0006	A0
0007	12
0008	00
0009	0F
000A	30
000B	90
000C	FD
000D	80
000E	F2
000F	78
0010	FF
0011	D8
0012	FE
0013	22
0014	00

Observando-se o Programa Fonte e os Códigos gerados pelo Compilador/Linker do Pinnacle, a última Instrução (RET) tem Código Binário 22 e ocupa a Posição 0013 da Memória de Programa.

```
                ORG 0
                CLR A
LOOP:           JB P1.0,$
                INC A
                MOV P2,A
                CALL ATRASO
                JNB P1.0,$
                SJMP LOOP

;*****************************

ATRASO:         MOV R0,#0FFH
                DJNZ R0,$
                RET
;*****************************
```

Addr	Opcodes	ASC	Label	Disassembly
0000	E4	ä		CLR A
0001	20 90 FD	‼ý	LOOP	JB P1.0,LOOP
0004	04	‼		INC A
0005	F5 A0	õ		MOV P2,A
0007	12 00 0F	‼‼		LCALL ATRASO
000A	30 90 FD	0‼ý		JNB P1.0,000A
000D	80 F2	€ò		SJMP LOOP
000F	78 FF	xÿ	ATRASO	MOV R0,#0FFh
0011	D8 FE	Øþ		DJNZ R0,0011
0013	22	"	(EXT1 INT)	RET
0014	00	‼		NOP
0015	00	‼		NOP
0016	00	‼		NOP
0017	00	‼		NOP
0018	00	‼		NOP

- Após a última Instrução, do endereço 0014 em diante, a Memória de Programa contém 00.
- Na prática, ela pode conter resíduo de outros programas gravados anteriormente.

Este programa exemplo ocupa 20 Bytes da Memória de Programa, ou seja, o Tamanho deste programa é de 20 Bytes.

20 Bytes

Addr	Opcodes	ASC	Label	Disassembly
0000	E4	ä		CLR A
0001	20 90 FD	‼ý	LOOP	JB P1.0,LOOP
0004	04	‼		INC A
0005	F5 A0	õ		MOV P2,A
0007	12 00 0F	‼‼		LCALL ATRASO
000A	30 90 FD	0‼ý		JNB P1.0,000A
000D	80 F2	€ò		SJMP LOOP
000F	78 FF	xÿ	ATRASO	MOV R0,#0FFh
0011	D8 FE	Øþ		DJNZ R0,0011
0013	22	"	(EXT1 INT)	RET
0014	00	‼		NOP
0015	00	‼		NOP
0016	00	‼		NOP
0017	00	‼		NOP
0018	00	‼		NOP

0000	E4
0001	20
0002	90
0003	FD
0004	04
0005	F5
0006	A0
0007	12
0008	00
0009	0F
000A	30
000B	90
000C	FD
000D	80
000E	F2
000F	78
0010	FF
0011	D8
0012	FE
0013	22
0014	00

20 Bytes

0013(hexa) = 20 (decimal)

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

Exemplo:

Campo do Rótulo (Label)

```
;*******************************************************************************
; Programa de debounce de chave mecânica
; Uma sub-rotina de atraso deve ser inserida para garantir o tempo de
; estabilização mecânica da chave. Para a maioria das chaves um atraso de
; 500 us é suficiente.
;*******************************************************************************

                ORG     0
                CLR     A               ; Inicializa o contador em zero
LOOP:           JB      P1.0,$          ; Testa se a chave está aberta
                INC     A               ; Incrementa o contador
                MOV     P2,A            ; Mostra o valor do contador na Porta P2
                CALL    ATRASO          ; Realiza o debounce da chave
                JNB     P1.0,$          ; Verifica se a chave continua fechada
                SJMP    LOOP            ; Retorna para nova contagem de pulso

;*******************************************************************************
```

FORMATO DO PROGRAMA FONTE

As declarações do programa fonte são constituídas pelos seguintes campos:

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

2) Campo da Operação :

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

Exemplo:

Campo da Operação (Mnemônicos)

```
;*****************************************************************
; Programa de debounce de chave mecânica
; Uma sub-rotina de atraso deve ser inserida para garantir o tempo de
; estabilização mecânica da chave. Para a maioria das chaves um atraso de
; 500 us é suficiente.
;*****************************************************************

            ORG     0
            CLR     A           ; Inicializa o contador em zero
LOOP:       JB      P1.0,$      ; Testa se a chave está aberta
            INC     A           ; Incrementa o contador
            MOV     P2,A        ; Mostra o valor do contador na Porta P2
            CALL    ATRASO      ; Realiza o debounce da chave
            JNB     P1.0,$      ; Verifica se a chave continua fechada
            SJMP    LOOP        ; Retorna para nova contagem de pulso

;*****************************************************************
```

FORMATO DO PROGRAMA FONTE

As declarações do programa fonte são constituídas pelos seguintes campos:

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

3) Campo do Operando:

- especifica o dado a ser operado pela instrução.

Exemplo:

Campo do Operando

```
;*****************************************************************
; Programa de debounce de chave mecânica
; Uma sub-rotina de atraso deve ser inserida para garantir o tempo de
; estabilização mecânica da chave. Para a maioria das chaves um atraso de
; 500 us é suficiente.
;*****************************************************************

            ORG     0
            CLR     A           ; Inicializa o contador em zero
LOOP:       JB      P1.0,$      ; Testa se a chave está aberta
            INC     A           ; Incrementa o contador
            MOV     P2,A        ; Mostra o valor do contador na Porta P2
            CALL    ATRASO      ; Realiza o debounce da chave
            JNB     P1.0,$      ; Verifica se a chave continua fechada
            SJMP    LOOP        ; Retorna para nova contagem de pulso

;*****************************************************************
```

FORMATO DO PROGRAMA FONTE

As declarações do programa fonte são constituídas pelos seguintes campos:

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

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:

Campo do Comentário

```
;*****************************************************************
; Programa de debounce de chave mecânica
; Uma sub-rotina de atraso deve ser inserida para garantir o tempo de
; estabilização mecânica da chave. Para a maioria das chaves um atraso de
; 500 us é suficiente.
;*****************************************************************

            ORG     0
            CLR     A               ; Inicializa o contador em zero
LOOP:       JB      P1.0,$          ; Testa se a chave está aberta
            INC     A               ; Incrementa o contador
            MOV     P2,A            ; Mostra o valor do contador na Porta P2
            CALL    ATRASO          ; Realiza o debounce da chave
            JNB     P1.0,$          ; Verifica se a chave continua fechada
            SJMP    LOOP            ; Retorna para nova contagem de pulso

;*****************************************************************
```

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.

Exemplo:

```
;*************************************************************************
; Programa: Contador de pulsos com debounce de chave mecânica.
; Uma sub-rotina de atraso deve ser inserida para garantir o tempo de
; estabilização mecânica da chave. Para a maioria das chaves um atraso de
; 500 us é suficiente.
;*************************************************************************

            ORG     0
            MOV     A,#00000000b ; Inicializa o contador em zero
LOOP:       JB      P1.0,$       ; Testa se a chave está aberta
            INC     A            ; Incrementa o contador
            MOV     P2,A         ; Mostra o valor do contador na Porta P2
            CALL    ATRASO       ; Realiza o debounce da chave
            JNB     P1.0,$       ; Verifica se a chave continua fechada
            MOV     P3,#'H'      ; Envia o Caracter H para a Porta P3
            SJMP    LOOP         ; Retorna para nova contagem de pulso

;*************************************************************************

ATRASO:     MOV R0,#0FFH    ; Valor do atraso para debounce da chave de
            DJNZ R0,$       ; aproximadamente 500 us
            RET
;*************************************************************************
```

TIPOS DE INFORMAÇÕES NO CAMPO DO OPERANDO

2. **Constante numérica –**

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

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

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

```
;*************************************************************************
; Programa: Contador de pulsos com debounce de chave mecânica.
; Uma sub-rotina de atraso deve ser inserida para garantir o tempo de
; estabilização mecânica da chave. Para a maioria das chaves um atraso de
; 500 us é suficiente.
;*************************************************************************

            ORG     0
            MOV     A,#00000000b ; Inicializa o contador em zero
LOOP:       JB      P1.0,$       ; Testa se a chave está aberta
            INC     A            ; Incrementa o contador
            MOV     P2,A         ; Mostra o valor do contador na Porta P2
            CALL    ATRASO       ; Realiza o debounce da chave
            JNB     P1.0,$       ; Verifica se a chave continua fechada
            MOV     P3,#'H'      ; Envia o Caracter H para a Porta P3
            SJMP    LOOP         ; Retorna para nova contagem de pulso

;*************************************************************************

ATRASO:     MOV R0,#0FFH  ; Valor do atraso para debounce da chave de
            DJNZ R0,$     ; aproximadamente 500 us
            RET
;*************************************************************************
```

TIPOS DE INFORMAÇÕES NO CAMPO DO OPERANDO

2. **Constante numérica –**

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

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

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

```
;*****************************************************************************
; Programa: Contador de pulsos com debounce de chave mecânica.
; Uma sub-rotina de atraso deve ser inserida para garantir o tempo de
; estabilização mecânica da chave. Para a maioria das chaves um atraso de
; 500 us é suficiente.
;*****************************************************************************

            ORG     0
            MOV     A,#00000000b ; Inicializa o contador em zero
LOOP:       JB      P1.0,$       ; Testa se a chave está aberta
            INC     A            ; Incrementa o contador
            MOV     P2,A         ; Mostra o valor do contador na Porta P2
            CALL    ATRASO       ; Realiza o debounce da chave
            JNB     P1.0,$       ; Verifica se a chave continua fechada
            MOV     P3,#'H'      ; Envia o Caracter H para a Porta P3
            SJMP    LOOP         ; Retorna para nova contagem de pulso

;*****************************************************************************

ATRASO:     MOV R0,#0FFH    ; Valor do atraso para debounce da chave de
            DJNZ R0,$       ; aproximadamente 500 us
            RET
;*****************************************************************************
```

PSEUDO-INSTRUÇÕES OU DIRETIVAS DO ASSEMBLER

As diretivas não geram código de máquina!!.

São utilizadas para complementar as informações que permitam a montagem efetiva do programa.

- Indicar o Endereço Inicial do Programa.
- Reservar área de Dados
- Definir equivalência entre valores
- Etc…

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.

```
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.
```

Principais Diretivas:

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.

```
ORG 0010h
DB 05h, 0CFh
```

Exemplo:

Armazena na posição 0010h da Memória de Programa, o Byte 05h e na posição seguinte (0011h) o Byte CFh

Principais Diretivas:

2) Diretiva DB – Define Byte

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

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.

```
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)
```

Principais Diretivas:

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)
```

Principais Diretivas:

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.
- O valor pode ser um valor numérico ou uma expressão.

- Uma vez declarado o valor da variável este não poderá mudar.

Principais Diretivas:

4) Diretiva EQU (=) – (Equate) Igual

Variable* EQU *value
Variable* = *value

Exemplo:

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

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
```