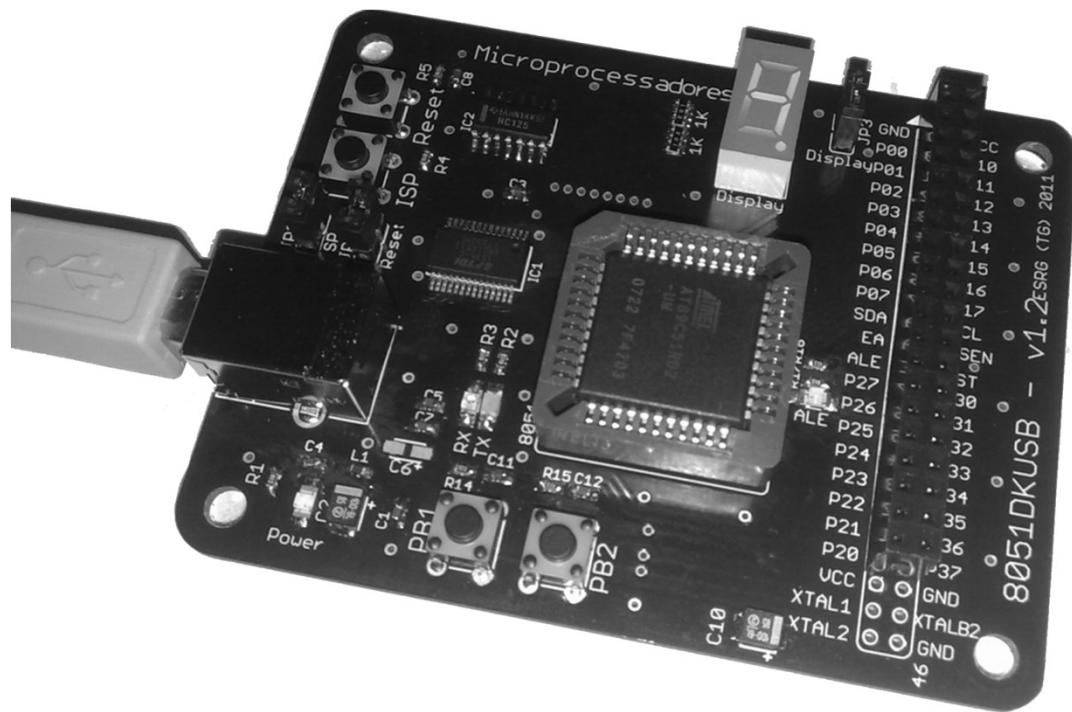
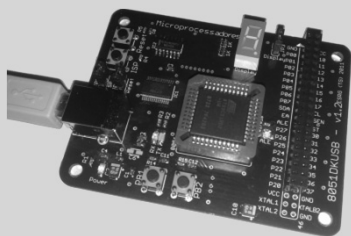


Mestrado Integrado em Eng. Electrónica Industrial e Computadores



Algoritmia

Microcontroladores
2º Ano – A04



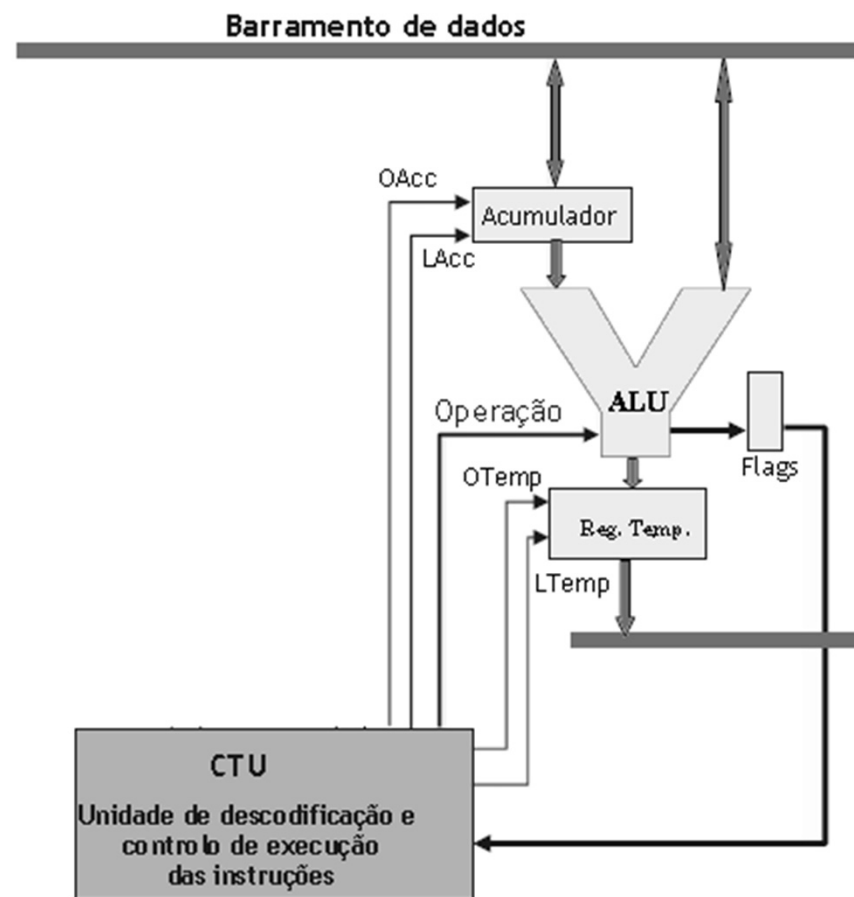
ALU

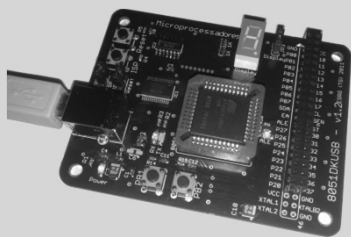
O registo de trabalho da ALU (Acumulador) é necessário para permitir que a ALU realize operações sobre dois operandos em simultâneo: dados do barramento e registo A;

Pelo mesmo motivo é necessário um registo para armazenar temporariamente o resultado da ALU.

Era possível armazenar o resultado no Acc?

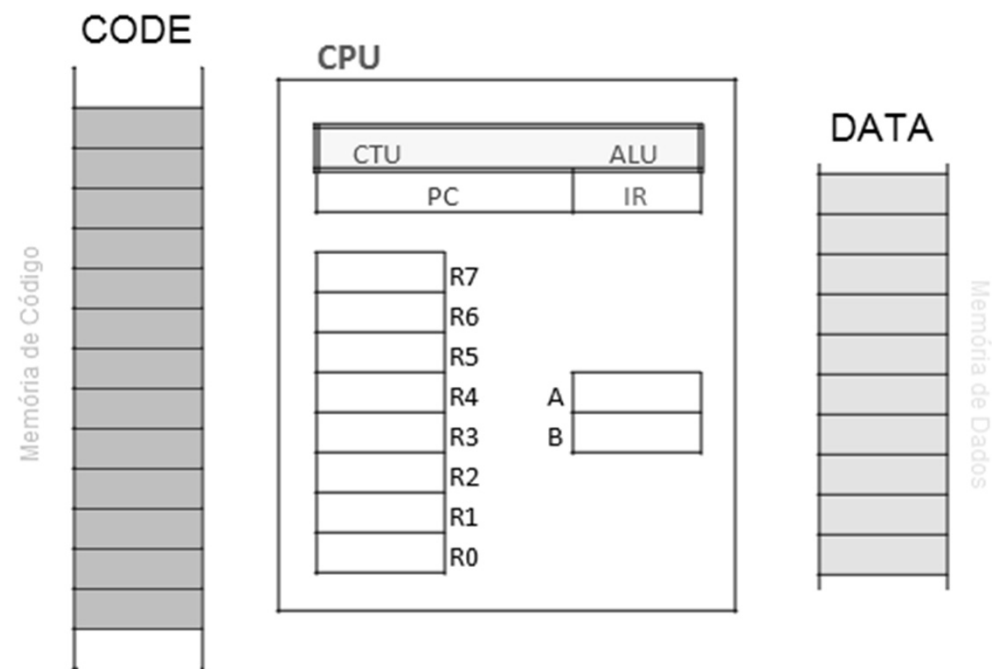
Flags: Carry, Zero e Paridade

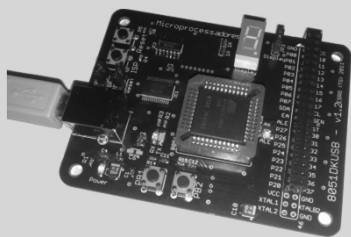




CPU ...

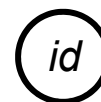
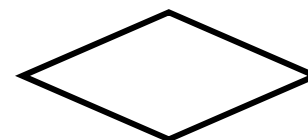
- CODE: memória onde são guardadas as instruções/programa. Não volátil;
- Qual o tamanho máximo da CODE?
- Memória de dados para armazenar as variáveis, o seu tamanho é fixo e igual a 256 bytes.
- Registo Acumulador – A – registo de trabalho da ALU, único endereçável ao bit;
- Registo B – registo extra da ALU;
- 8 registos de propósito geral R0 a R7;
- Flags: CY – carry e Z – zero
- A memória de dados só pode ser endereçada utilizando os registos R0 e R1.



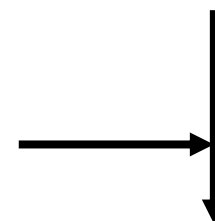


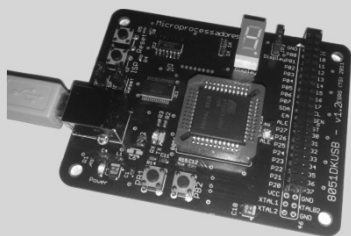
Alguns símbolos do fluxograma

- **Bloco de Decisão**
- **Bloco de processamento**
- **Terminal do programa**
- **Subrotina**
- **Bloco entrada/saída**
- **Seta de fluxo do programa**
- **Ponto de ligação**

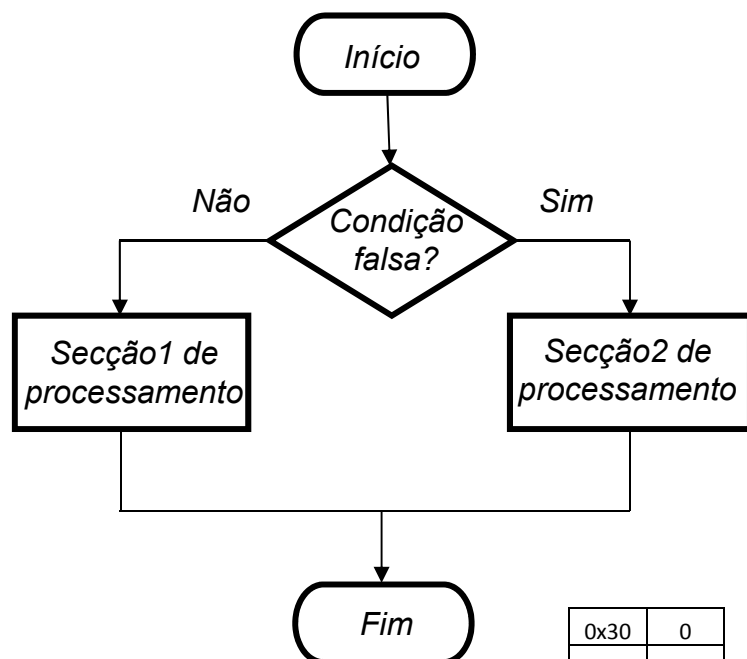


*Setas de
ligação*





Estruturas do IF/THEN/ELSE



Exercício: Converta um dígito armazenado no endereço apontado por R0 para um caracter ASCII que representa o seu valor hexadecimal.

O valor apontado por R0 contém apenas um dígito hexadecimal (os quatro bits mais significativos são 0). Guarde o resultado no registo B.

Exemplo:

Entrada: (R0) = 0Ch

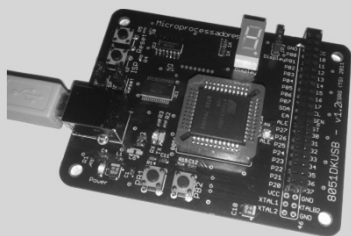
Saída: (B) = 43h = 'C'

Entrada: (R0) = 06h

Saída: (B) = 36h = '6'

0x30	0
0x31	1
0x32	2
0x33	3
0x34	4
0x35	5
0x36	6
0x37	7
0x38	8
0x39	9

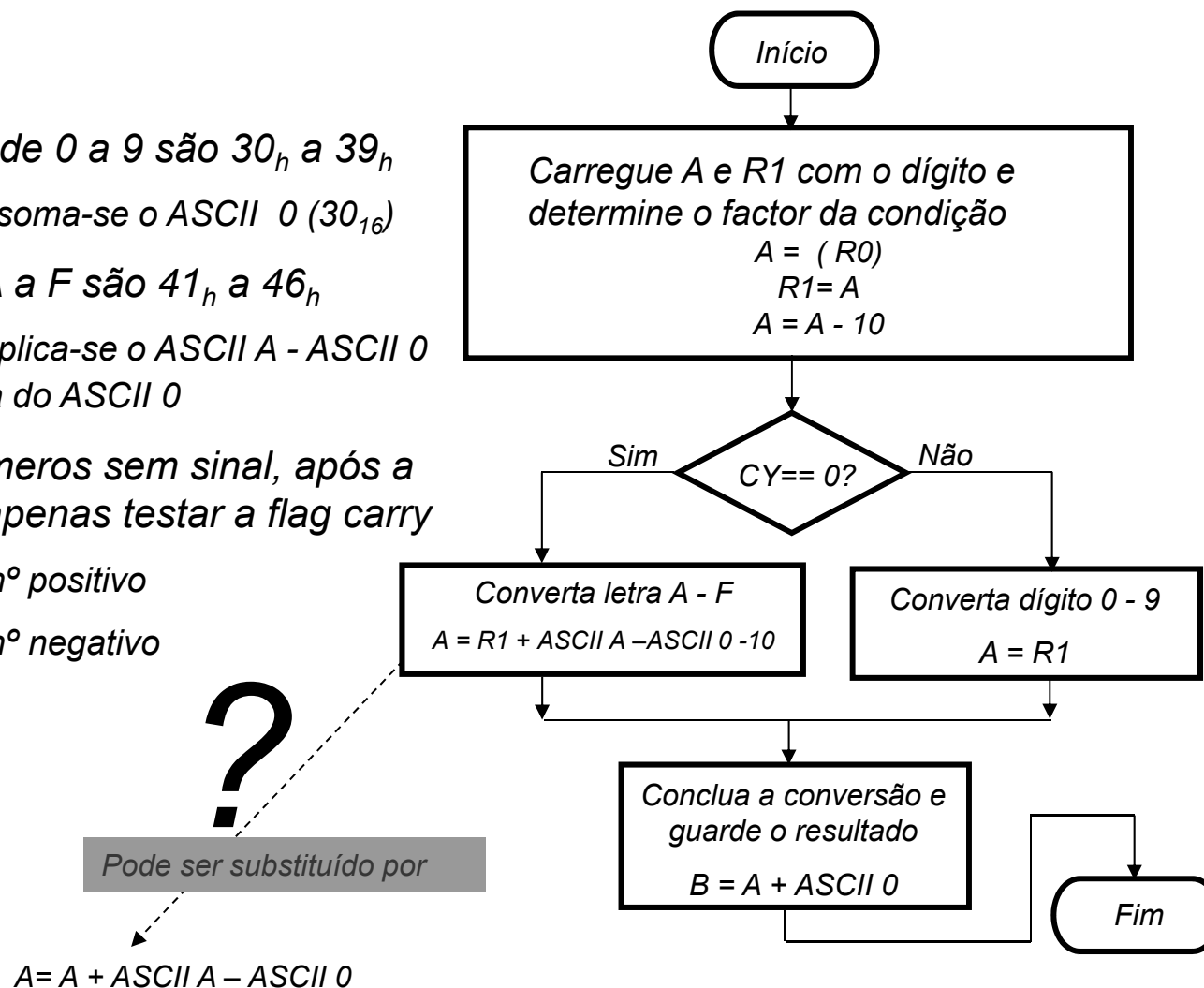
0x41	A
0x42	B
0x43	C
0x44	D
0x45	E
0x46	F

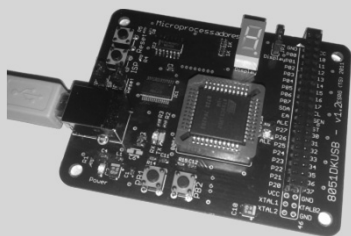


Estruturas do IF/THEN/ELSE

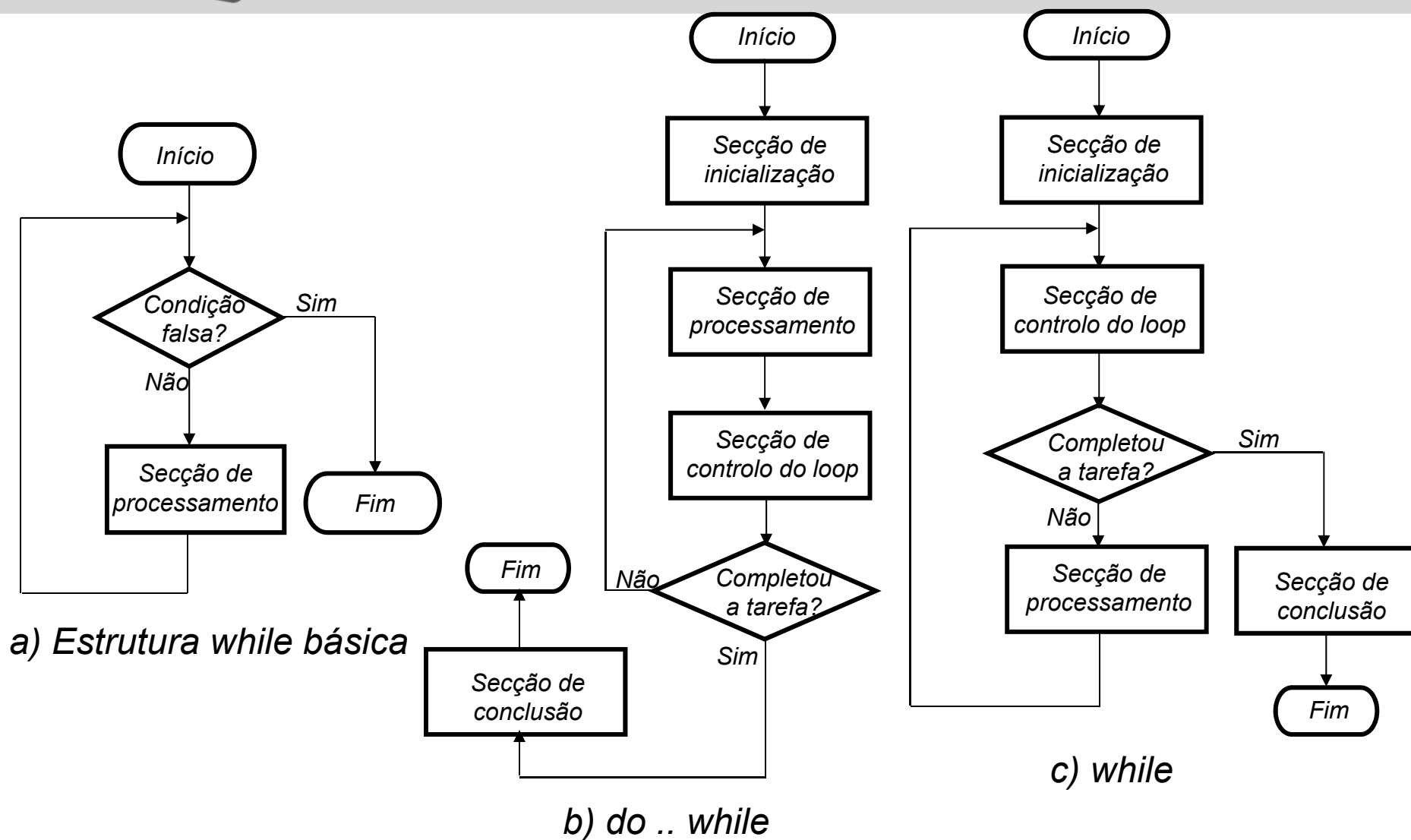
Análise do Problema:

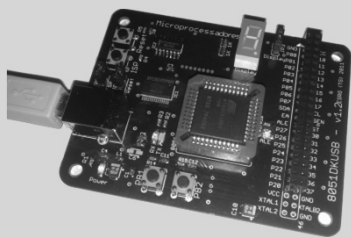
- Os ASCII dos dígitos de 0 a 9 são 30_h a 39_h
 - Para estes dígitos soma-se o ASCII 0 (30_{16})
- Os ASCII das letras A a F são 41_h a 46_h
 - Para estas letras aplica-se o ASCII A - ASCII 0 - 10 antes da soma do ASCII 0
- Como se trata de números sem sinal, após a subtração pode-se apenas testar a flag carry
 - CY = 0, resultado nº positivo
 - CY = 1, resultado nº negativo





Estruturas do Loop





Estruturas do Loop

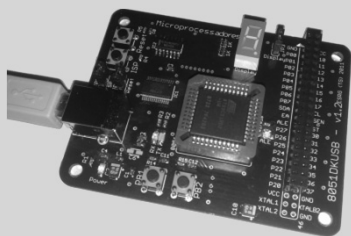
- Exercício: Calcular a soma de um conjunto de números armazenados consecutivamente na RAM interna a partir do endereço armazenado em R0, (R0). O endereço do último número é dado pelo (40H). Considere que a soma é sempre inferior a 256 e guarde o resultado em 40H.

Análise do Problema:

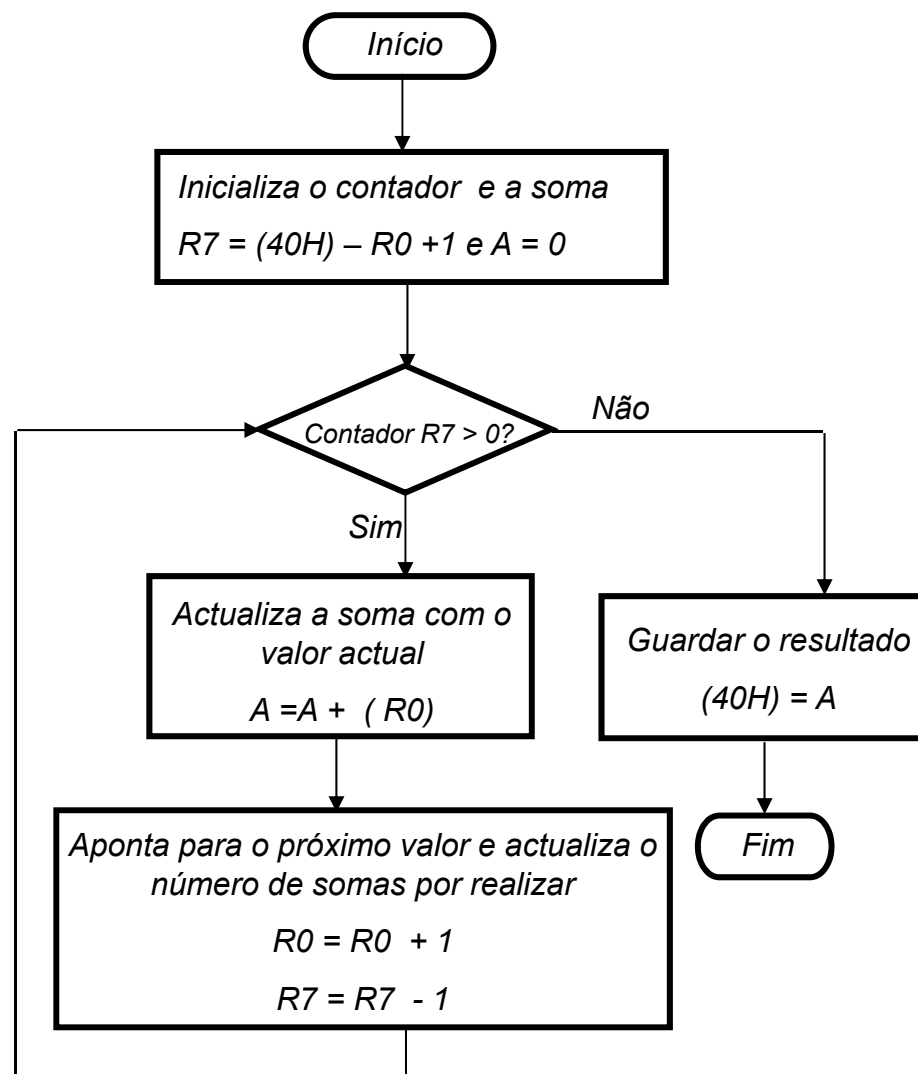
- Exemplo:

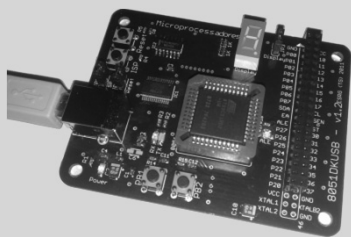
Entrada:	(R0)	= 4
	(R0 +1)	= 5
	(R0 +2)	= 65
	(R0 +3)	= 70
	(40H)	= (R0 +3)
• Saída:	(40H)	= 144

- *Número de elementos a somar*
 - $(40H) - (R0) + 1$
- *Como a soma é inferior a 256 pode-se efectuar todo o cálculo a partir do acumulador*
- *Como os dados estão consecutivamente armazenados pode-se efectuar um loop usando o contador e endereçamento indirecto a partir de R0 para aceder aos números*



Estruturas do Loop





Estruturas do Loop

- Exercício: Determinar o índice do caracter **z** numa *string* terminada pelo caracter *Carriage Return*. O endereço da *string* é dado pelo (R0) e o resultado será guardado no acumulador B. Na ausência do caracter pesquisado (B) = 0FFh.

Análise do Problema:

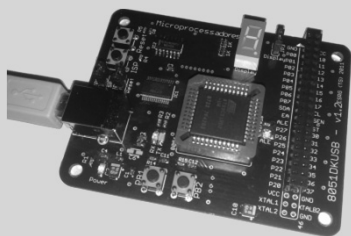
- Exemplo:

Entrada:	(R0)	= 'g'
	(R0 +1)	= 'a'
	(R0 +2)	= 'z'
	(R0 +3)	= 'e'
	(R0 +4)	= 'l'
	(R0 +5)	= 'a'
	(R0 +6)	= 0Dh
• Saída:	B	= 2

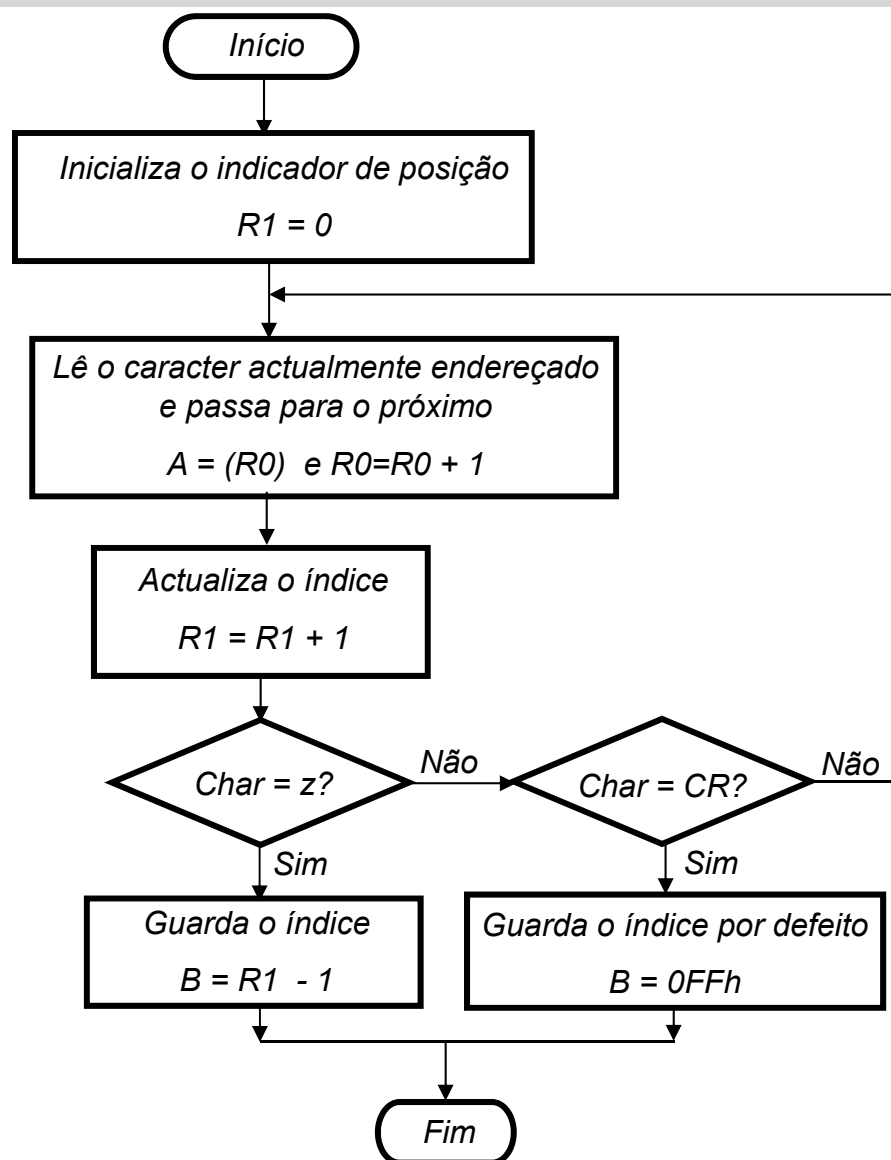
•A pesquisa do caractere será realizada até se encontrar o caracter **z** ou o CR = 0Dh. Pelo que a condição de paragem será

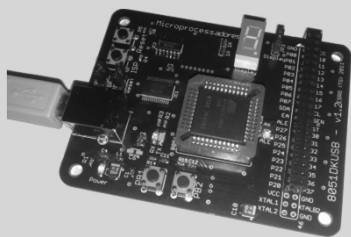
•char==CR || char=='z'

•Pode-se usar o *do ... while* dado que o único processamento após o teste da condição de paragem consiste no incremento do indicador da posição. No entanto, a solução com o *while ... do* seria a melhor. Porquê?



Estruturas do Loop





Estruturas do Loop

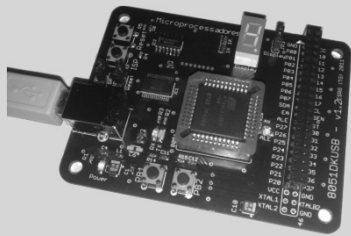
- Exercício: Determinar o número de elementos nulos, positivos e negativos de uma série (comprimento em R1) de números com sinal de 16 *bits* armazenados consecutivamente a partir do endereço dado por R0. Os resultados serão armazenados em R2, R3 e R4 para o número de negativos, de zeros e de positivos, respectivamente.

Análise do Problema:

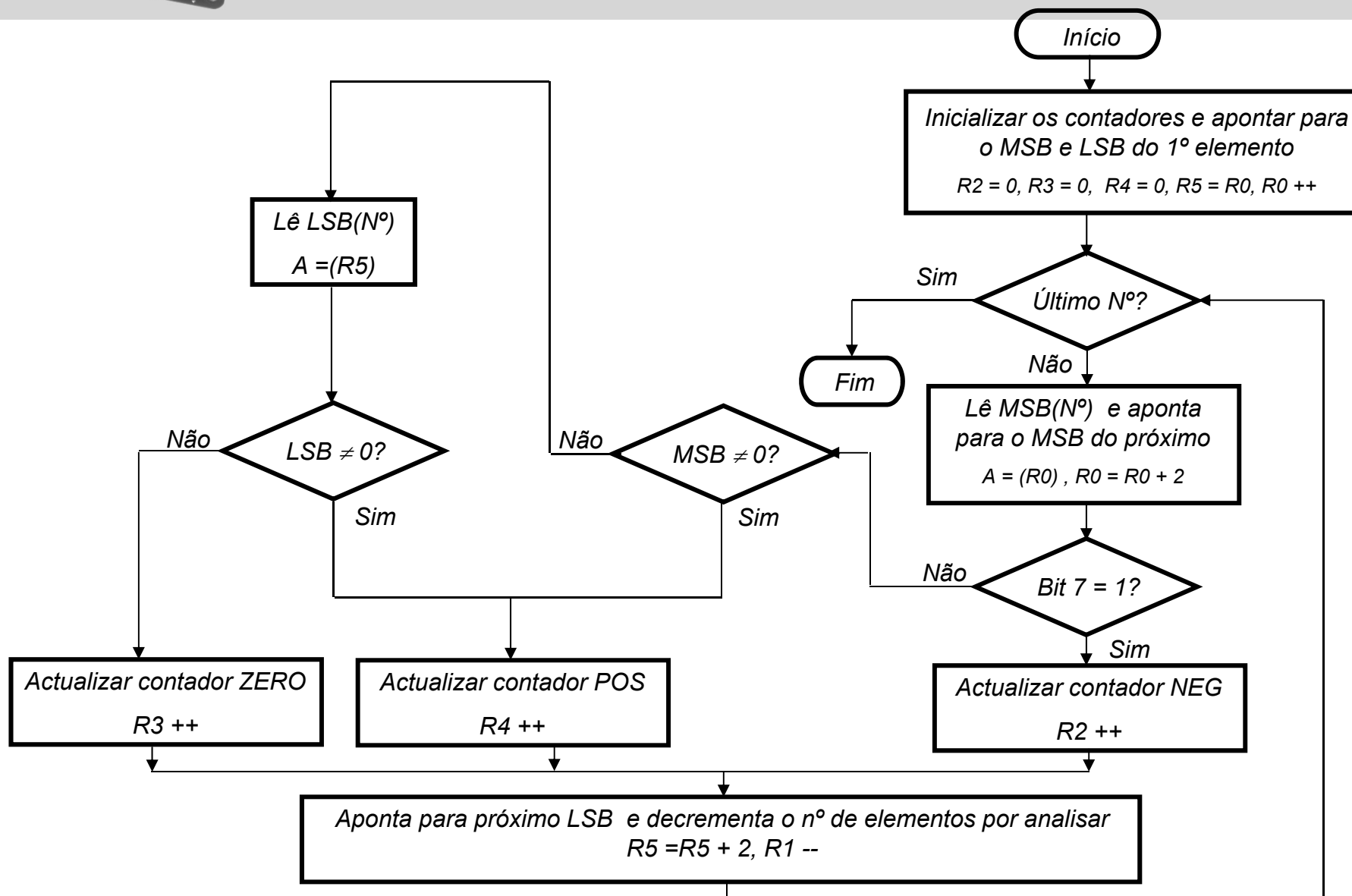
- Exemplo:

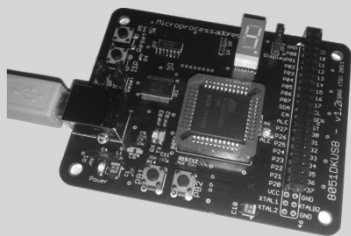
Entrada:	(R0)	= 7602h
	(R0 +1)	= 8d48h
	(R0 +2)	= 2120h
	(R0 +3)	= 0000h
	(R0 +4)	= E605h
	(R0 +5)	= 0004h
Saída:	R2	= 2
	R3	= 1
	R4	= 3

- *Para determinar se o número é positivo ou negativo basta testar o bit mais significativo*
 - *Sendo os números de 16-bit apenas precisamos de testar o bit 7 do MSB*
 - *Bit 7 = 1 número negativo*
 - *Bit 7 = 0 número positivo*
- *Para se certificar se o número é nulo deve-se testar os 2 bytes*
- *O teste do bit 7 pode ser realizado com máscaras ou então rodando e testando apenas o bit*

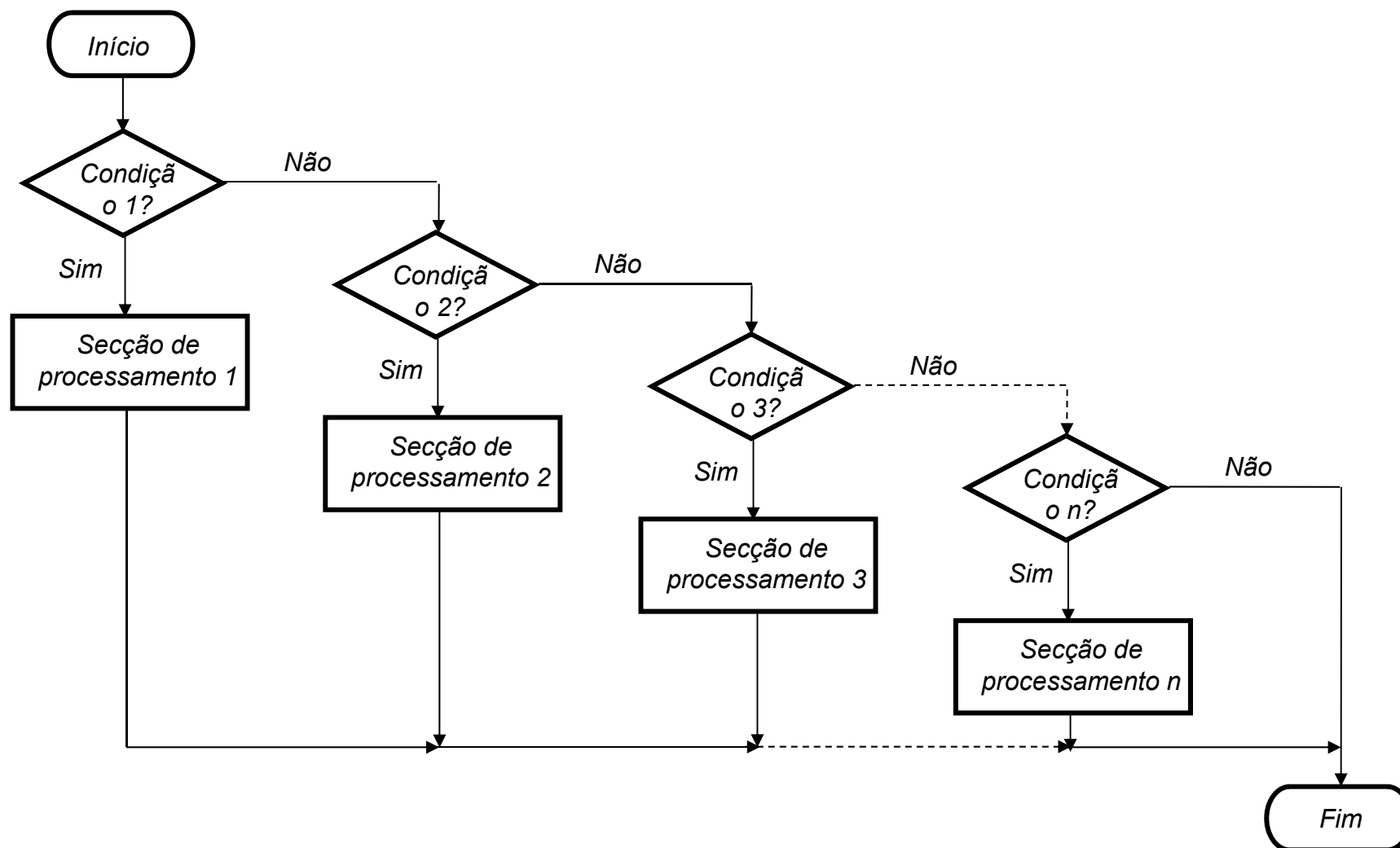


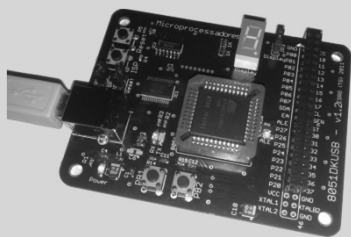
Estruturas do Loop





Estruturas do CASE





Estruturas do Case

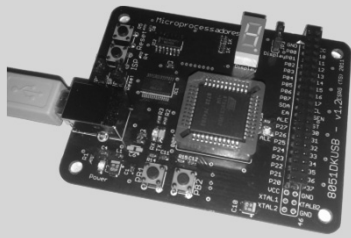
- Exercício: Contar a ocorrência de elementos nulos, pares e ímpares de uma série (comprimento em R1) de números de 8 bits armazenados consecutivamente a partir do endereço dado por R0. Os resultados serão armazenados em R2, R3 e R4 para o número de pares, de zero e de ímpares, respectivamente.

Análise do Problema:

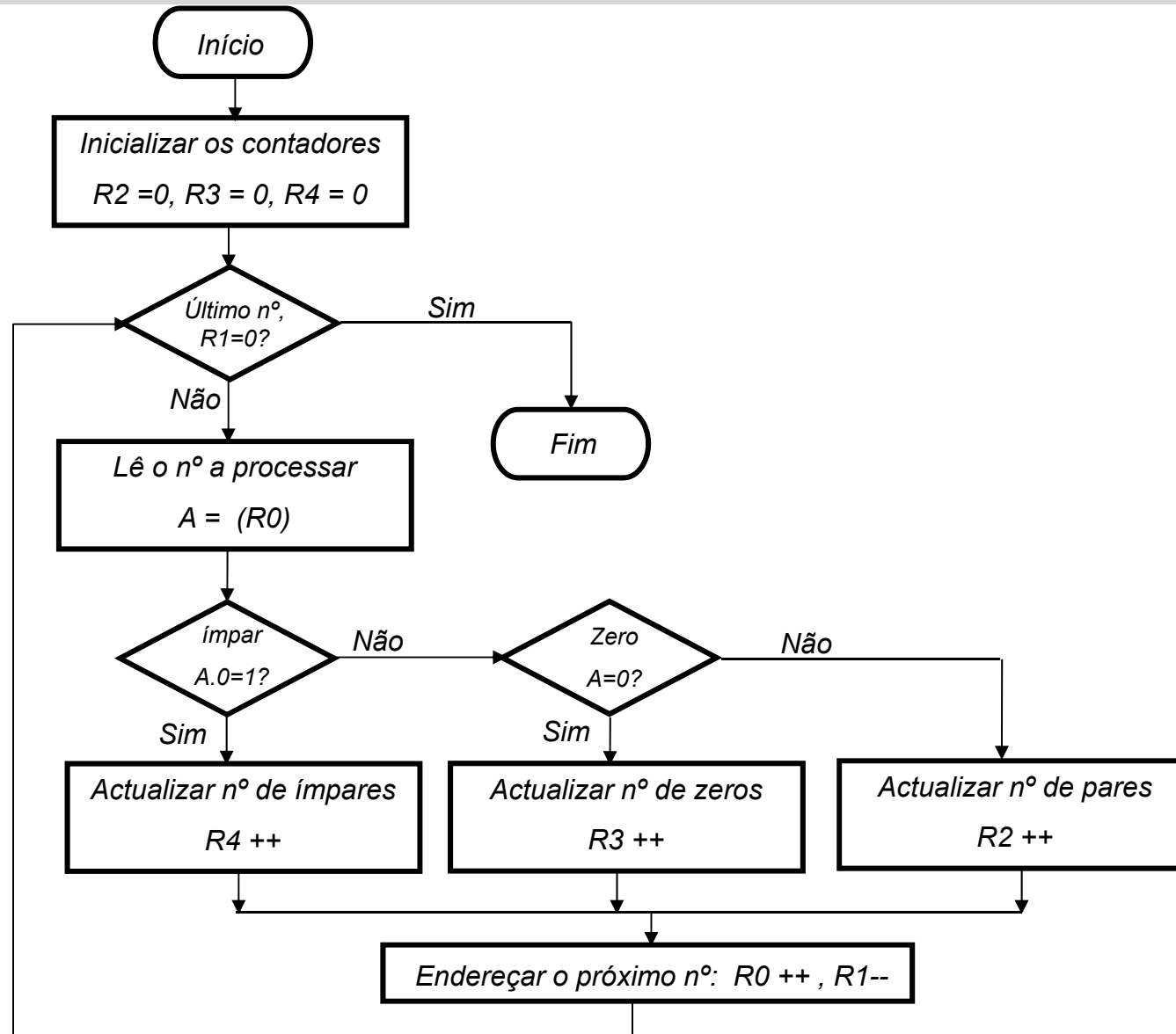
- Exemplo:

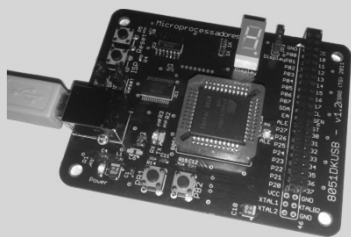
Entrada:	(R0)	= 02h
	(R0 +1)	= 48h
	(R0 +2)	= 20h
	(R0 +3)	= 00h
	(R0 +4)	= 05h
	(R0 +5)	= 00h
Saída:	R2	= 3
	R3	= 2
	R4	= 1

- Para determinar se o número é par ou ímpar basta testar o bit menos significativo
 - Bit 0 = 1 número ímpar
 - Bit 0 = 0 número par
- Para se certificar se o número é nulo deve-se testar todos os bits
- O teste do bit 0 pode ser realizado com máscaras ou então rodando e testando apenas o bit
- Alternativamente, pode-se testar directamente o bit 0



Estruturas do Case





Exercícios

- *Apresente um algoritmo para o seguinte problema: considere uma tabela armazenada a partir do endereço 20h da memória de dados. Determine os elementos que têm o maior e o menor número de bits a 1, respectivamente.*

Exemplos:

Entrada:

(20h) = 00001111b

(21h) = 01110101b

(22h) = 00011000b

(23h) = 10000000b

(24h) = 00010010b

(25h) = 10000111b

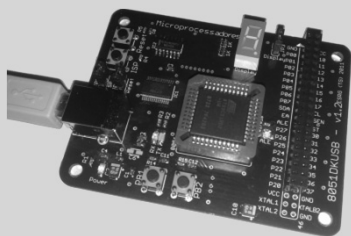
Saída:

*(R2)=01110101b tem cinco
1s*

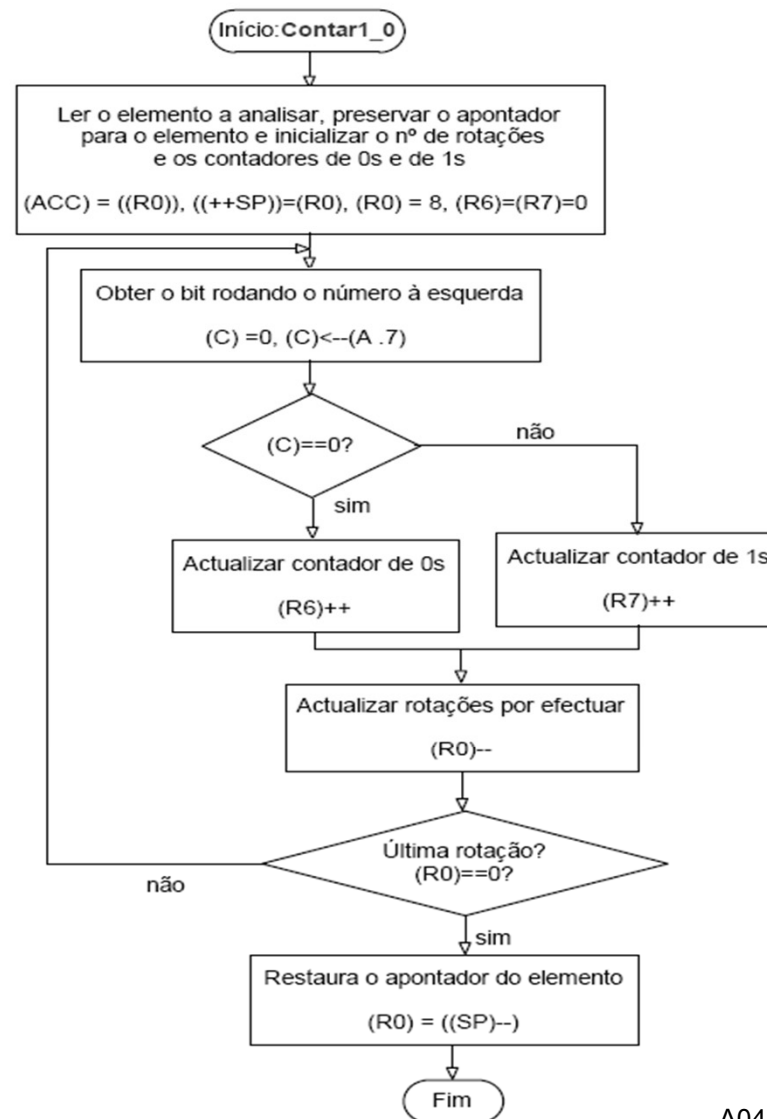
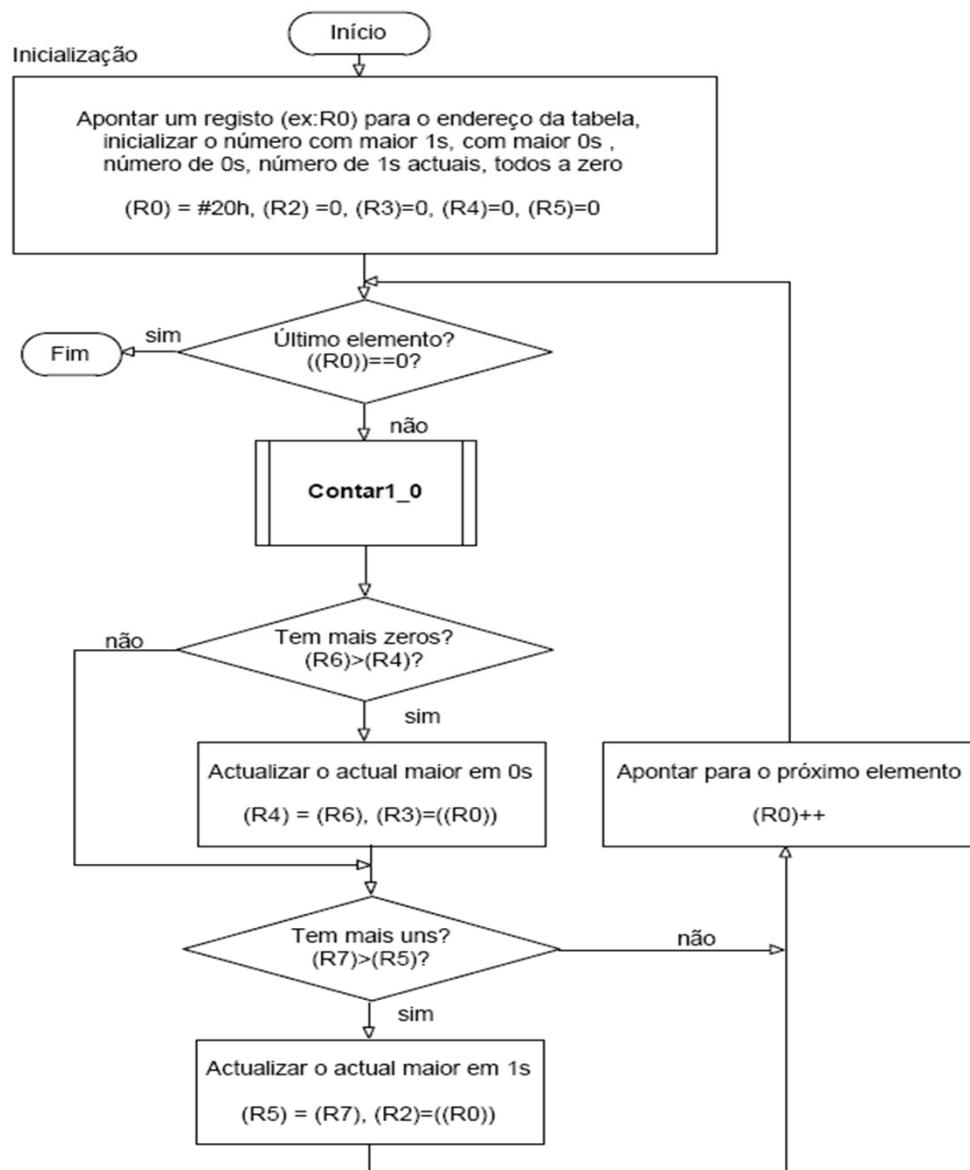
(R3)=10000000b tem sete 0s

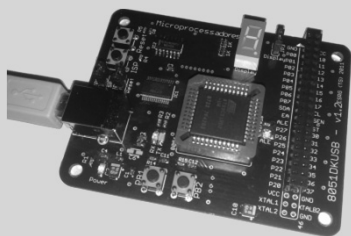
Análise do Problema

1. *Como não foi indicado o nº de elementos, considera-se que o nº zero indica o fim da tabela*
2. *Considera-se que os elementos são de 8-bits, pelo que seriam necessários 8 rotações para analisar cada elemento*
3. *Para cada elemento da tabela é invocada uma subrotina para a contagem dos 1s e 0s*
 - i. *O parâmetro é passado através do registo (R0:aponta para o elemento a analisar)*
 - ii. *No algoritmo podia-se usar R1 para a contagem das rotações, evitando a preservação de R0. Contudo, apenas para exemplificar/forçar o uso da pilha optou-se pela utilização de R0 como contador de rotações por realizar.*



Exercícios





Exercícios

- Dada uma matriz 3x3 determine qual a célula (i,j) do maior elemento. Considere que o registo R0 aponta para a célula (0,0) da matriz.

Exemplo:

Entrada:

33 55 44

34 58 88

36 10 12

Saída: (1,2) $\rightarrow M[1][2] = 88$

Entrada:

5 77 6

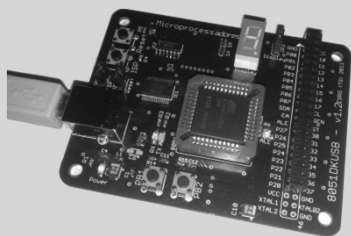
6 13 3

1 4 11

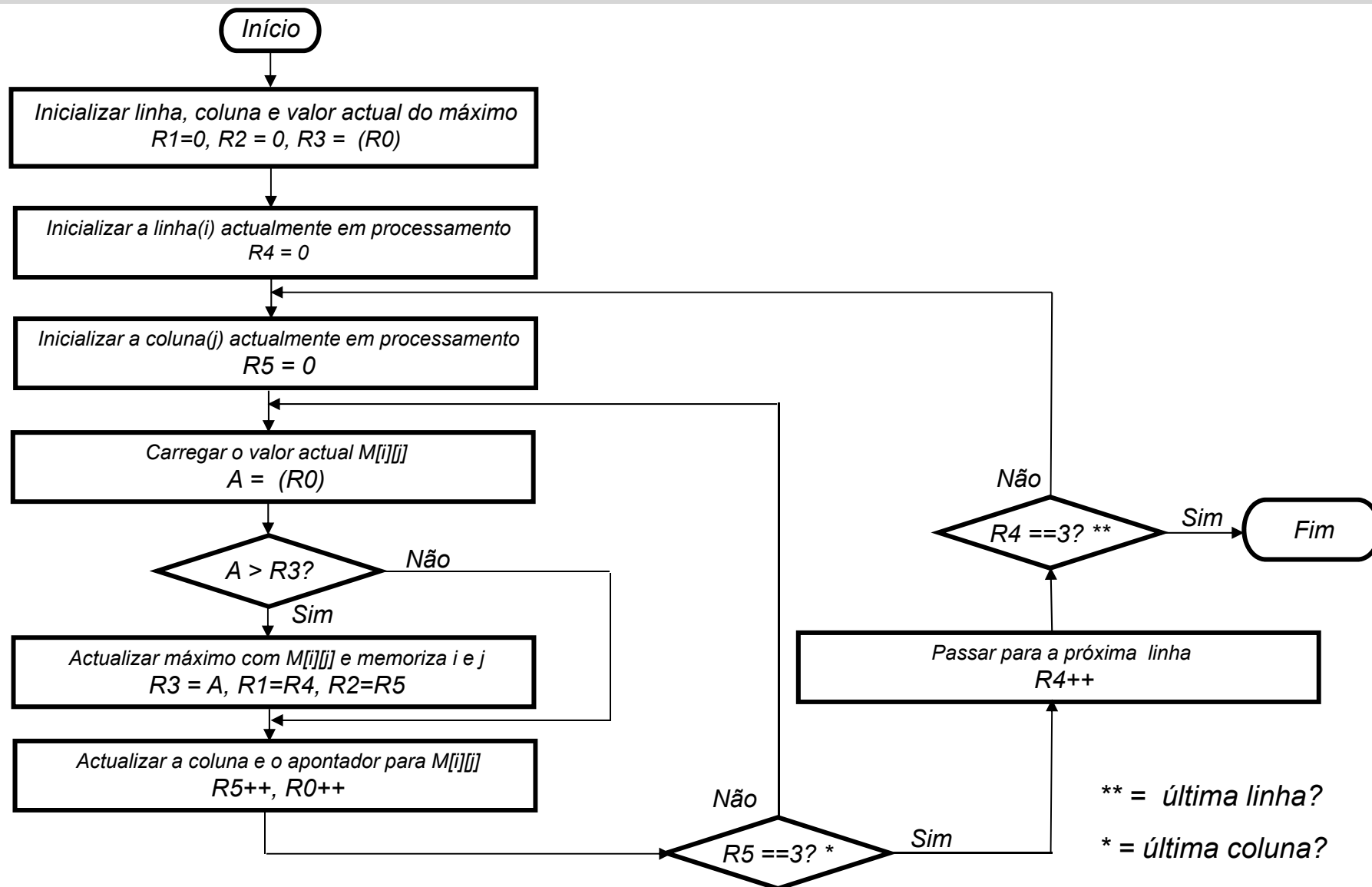
Saída: (0,1) $\rightarrow M[0][1]=77$

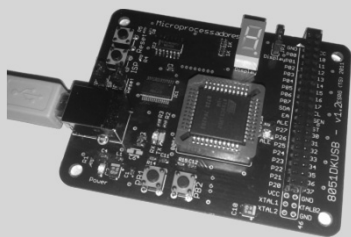
Análise do Problema

1. Vamos considerar que os valores nas células são inferiores a 256
2. A comparação do maior valor actual pode ser efectuado usando a subtracção e comparando o valor do Carry caso não exista uma instrução de comparação adequada
3. Não se esqueça que os elementos da matriz são armazenadas sequencialmente na memória
O apontador para a célula (i,j) = $(R0) + i*3 + j$



Exercícios





Exercícios

- *Apresente um algoritmo para o seguinte problema: copiar todos os números pares de uma tabela armazenada a partir do endereço 20h da memória de dados para a posição 61h, na posição 60h deverá ser armazenado o nº de pares.*

Exemplos:

Entrada:

(20h) = 00001111b
(21h) = 01110101b
(22h) = 00011000b
(23h) = 10000000b
(24h) = 00010010b
(25h) = 00000000b

Saída:

(60h) = 3
(61h) = 00011000b
(62h) = 10000000b
(63h) = 00010010b

Análise do Problema

1. *Como não foi indicado o nº de elementos, considera-se que o nº zero indica o fim da tabela*
2. *Para detectar que um nº é par basta testar o seu bit menos significativo.*