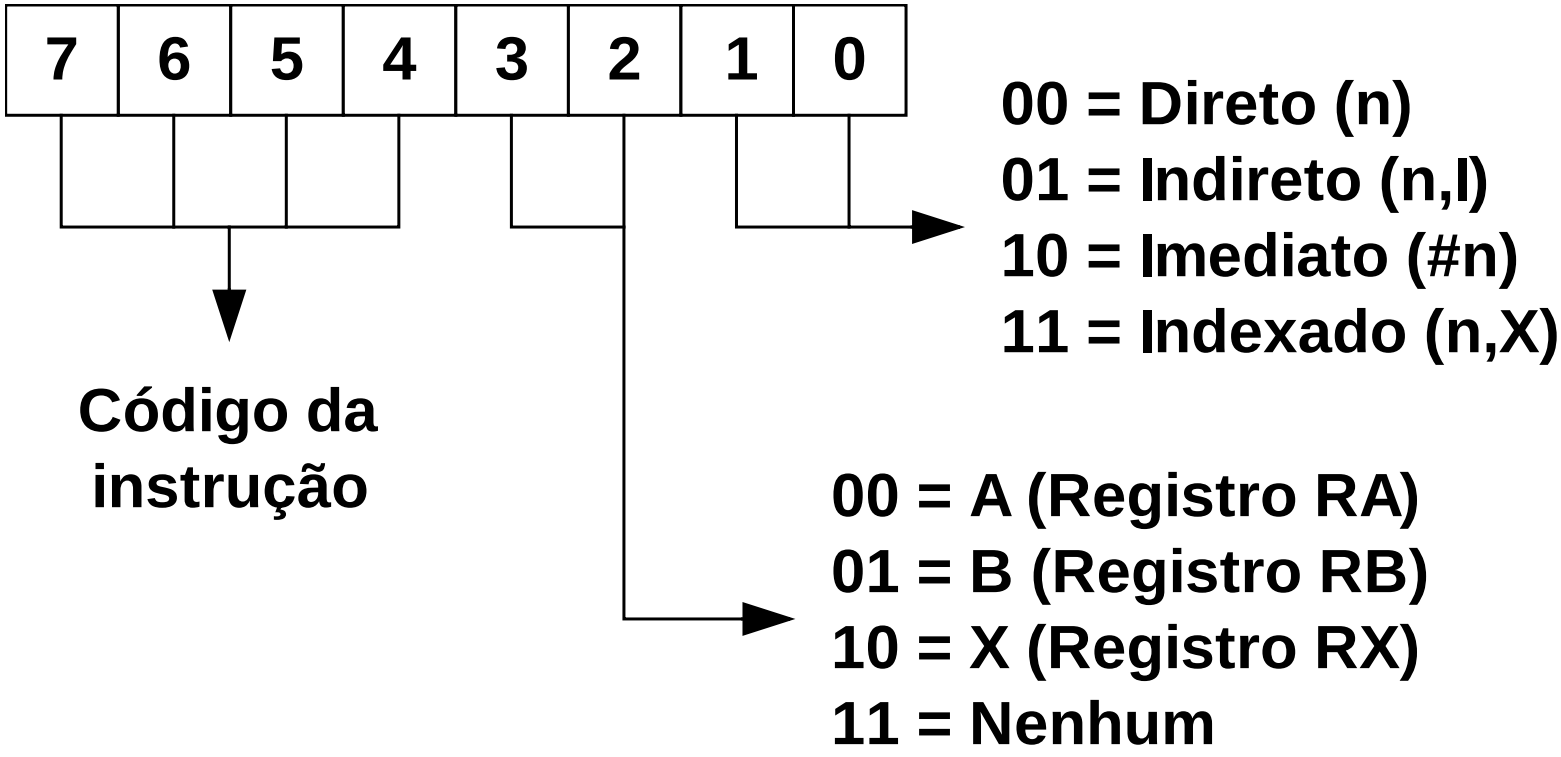
RAMSES =
NEANDER + Novas Instruções

Prof. Sérgio Luis Cechin

.inf INSTITUTO DE INFORMÁTICA UFRGS

Relembrando...



Relembrando...

- Modo **direto** - “n”
  - Compatível com o NEANDER
  - Ex: LDR A,NOME
- Modo **indireto** - “ n,I”
  - Ex: LDR A,NOME,I
- Modo **imediato** - “#n”
  - Ex: LDR A,#NOME
- Modo **indexado** - “n,X”
  - Ex: LDR A,NOME,X

Códigos de Condição

- Refletem o estado da última operação executada
- Não refletem o conteúdo atual de nenhum registrador
- N
  - Indica se o resultado da última instrução foi um número negativo (“interpretação” de complemento de 2)
- Z
  - Indica se o resultado da última instrução foi “zero”
- C
  - Seu significado depende da última instrução
  - As instrução são: ADD, SUB, NEG e SHR

Instruções

Gerenciamento (iguais ao NEANDER)

- NOP: Nenhuma operação
- HLT: Para a execução

Transferências

- LDR <*reg*> <*end*>

  reg ← MEM(end)

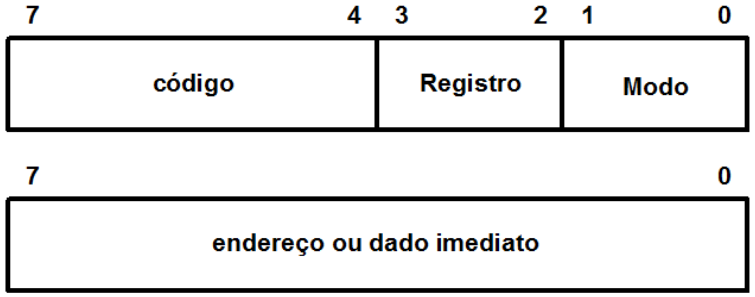
- STR <*reg*> <*end*>

  reg → MEM(end)

Exercício 1 (5min)

- Codificar as seguintes instruções

LDR A, 128
STR B, 129, I
LDR X, #130
LDR B, 131, X
STR X, 132, X



LDR = 0010
STR = 0001

A = 00
B = 01
X = 10

Direto = 00
Indireto = 01
Imediato = 10
Indexado = 11

Exercício 2 (5 min)

- Indique a sequência de leitura e escritas, a partir do “PC”, realizadas em cada uma das seguintes instruções

```
LDR     A, 128
STR     B, 129, I
LDR     X, #130
LDR     B, 131, X
STR     X, 132, X
```

Ex: A ← MEM(MEM(PC))

- Perceba que o único endereço necessário é o valor do PC!

Exercícios 3 (10 min)

- Implementar, em *assembly* do RAMSES, as seguintes atribuições "C" abaixo
- Utilize programação simbólica
- No caso de 16 bits, utilize o menor endereço para o byte menos significativo

```
char c;
unsigned char k;
unsigned short int i;

k = c;
k = i;
i = k;
```

Observe que:

char = 8 bits com sinal

unsigned char = 8 bits sem sinal

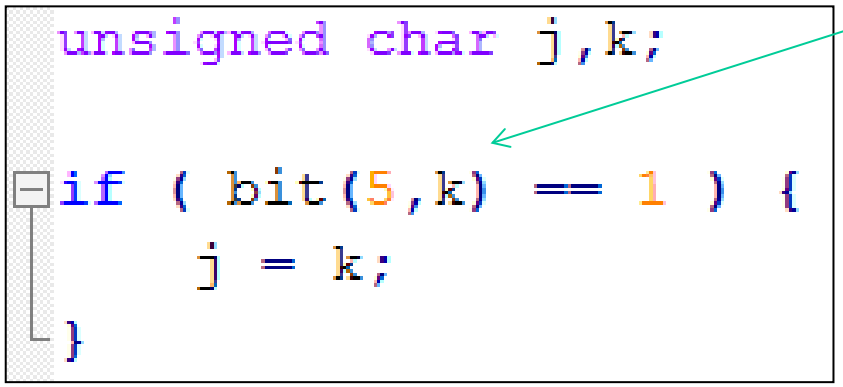
unsigned short int = 16 bits sem sinal

Lógicas
(Iguais ao NEANDER)

- OR <reg> <end>
- AND <reg> <end>
- NOT <reg>
- Onde
  - <reg> = A, B ou X
  - <end> = direto, indireto, imediato ou indexado

Exercício 4 (5 min)

- Implementar em *assembly* do RAMSES, o seguinte trecho em "C"



Observar a "criação" de uma função

Dica: usar máscara AND para isolar o bit 5

Aritméticas
(Igual ao NEANDER)

- **ADD <reg> <end>**
- Onde
  - <reg> = A, B ou X
  - <end> = direto, indireto, imediato ou indexado

Aritméticas (Novidade)

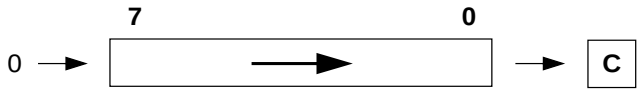
- **SUB** ***<reg> <end>***

  r ← r – MEM(end)

- **NEG** ***<reg>***

  r ← 0 – r

- **SHR** ***<reg>***



Exercício 5 (5 min)

- Implementar em *assembly* do RAMSES o seguinte trecho de código

```
char c;
short int x;


x = c;
```

sizeof(char) = 8
sizeof(short int) = 16

Como é implementada a atribuição quando:

- "c" é positivo?
- "c" é negativo?

Exercício 6 (10 min)

- Implementar em *assembly* do RAMSES o seguinte trecho de código

```
char x,y;

if ( x >= y ) {
     x = x - y;
}
```

Utilize SUB,
- para realizar a comparação
- para realizar a diferença

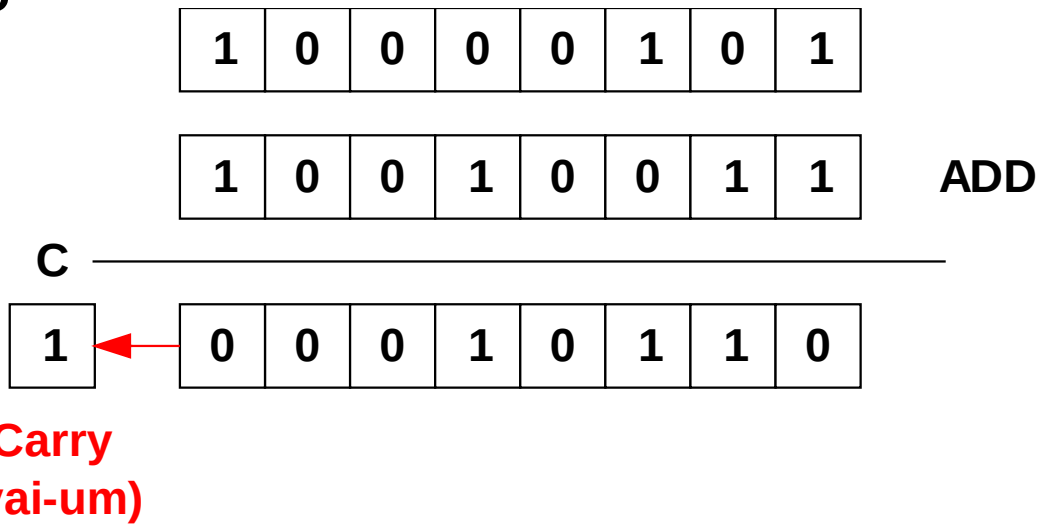
Para realizar a comparação de números **com sinal**, deve-se usar JN

Código de Condição “C”

- É gerado por quatro instruções:
  - ADD, SUB, NEG e SHR
- Instrução ADD
  - Corresponde ao “vai-um” do bit 7
- Instruções SUB e NEG
  - Corresponde ao “empresta um” do bit 7
- Instrução SHR
  - Corresponde ao bit 0

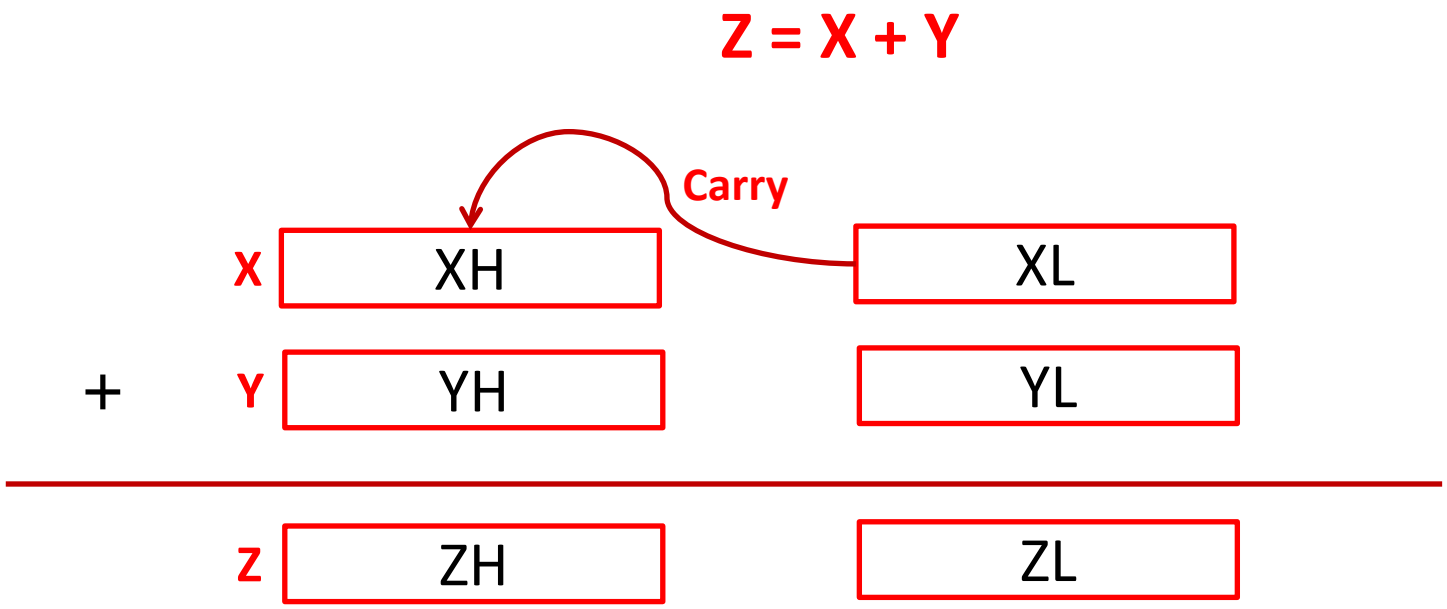
Exemplo – ADD

- Sequência de instruções
  - LDR A,#H85
  - ADD A,#H93



Exercício 7 (10 min)

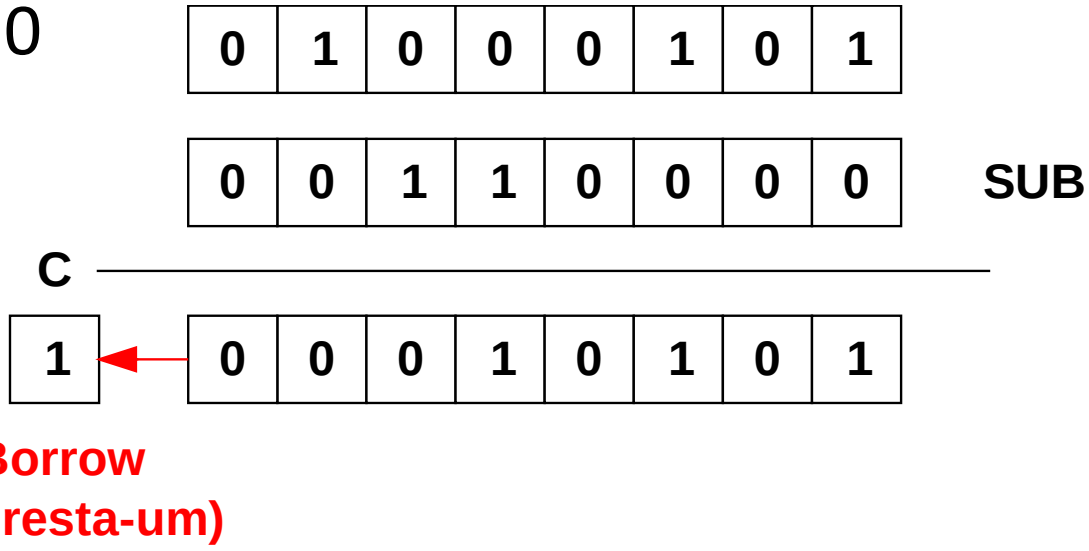
- Implementar em *assembly* do RAMSES a soma de dois números de 16 bits sem sinal



Exemplo – SUB

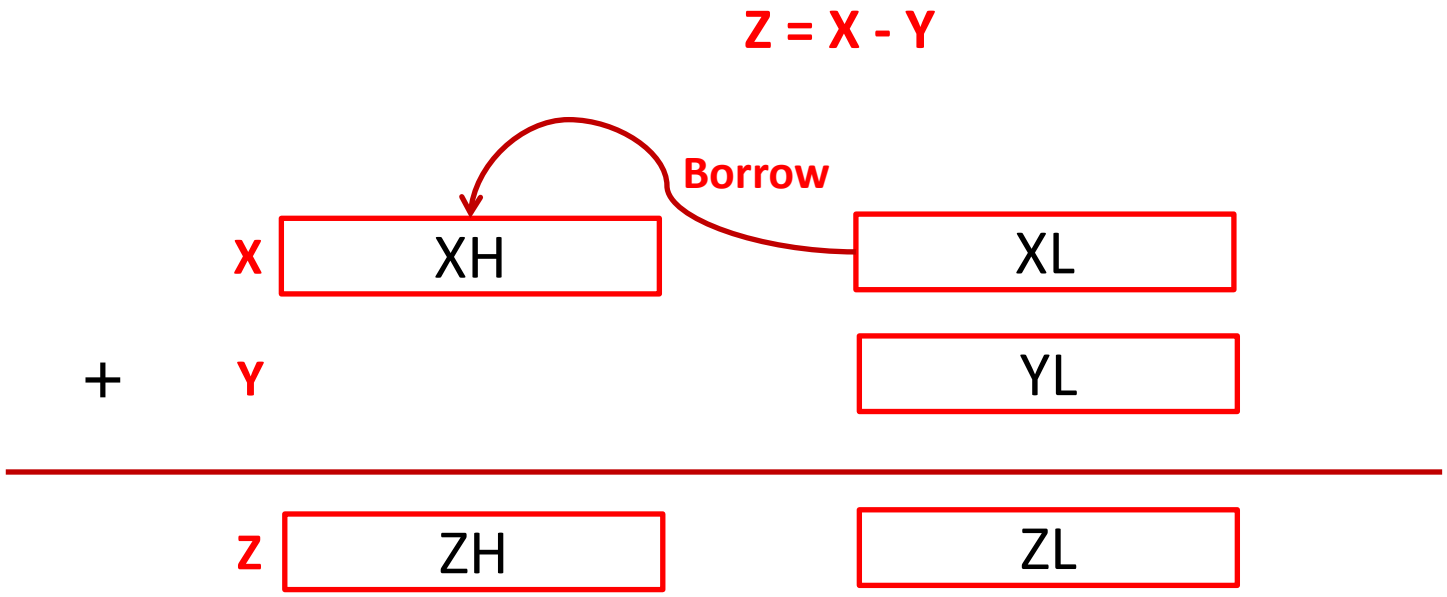
- Sequência de instruções

```
LDR     A,#H45
SUB     A,#H30
```



Exercício 8 (10 min)

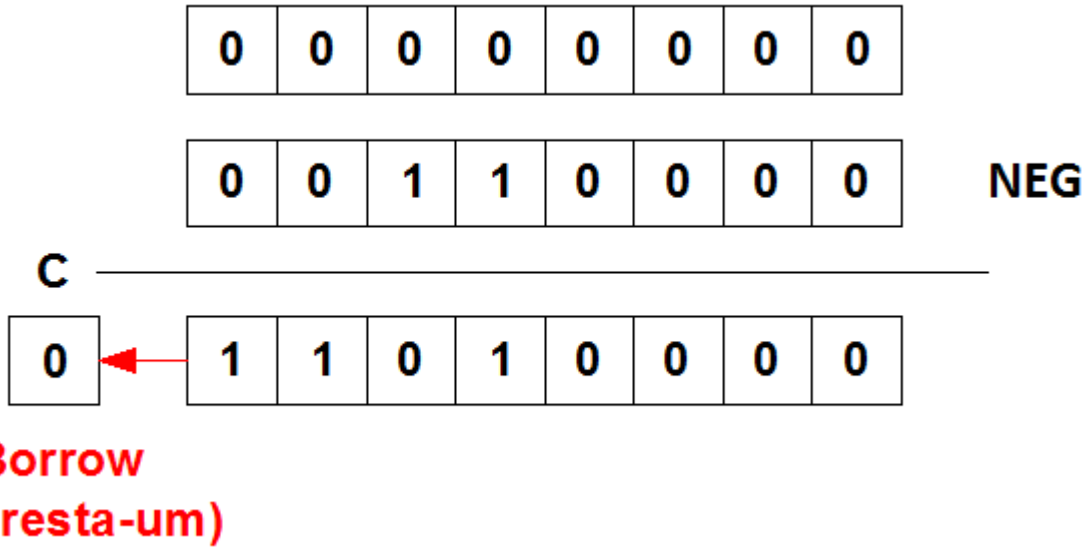
- Implementar em *assembly* do RAMSES a diferença de dois números: X com 16 bits e Y com 8 bits. Ambos sem sinal



Exemplo – NEG

- Sequência de instruções

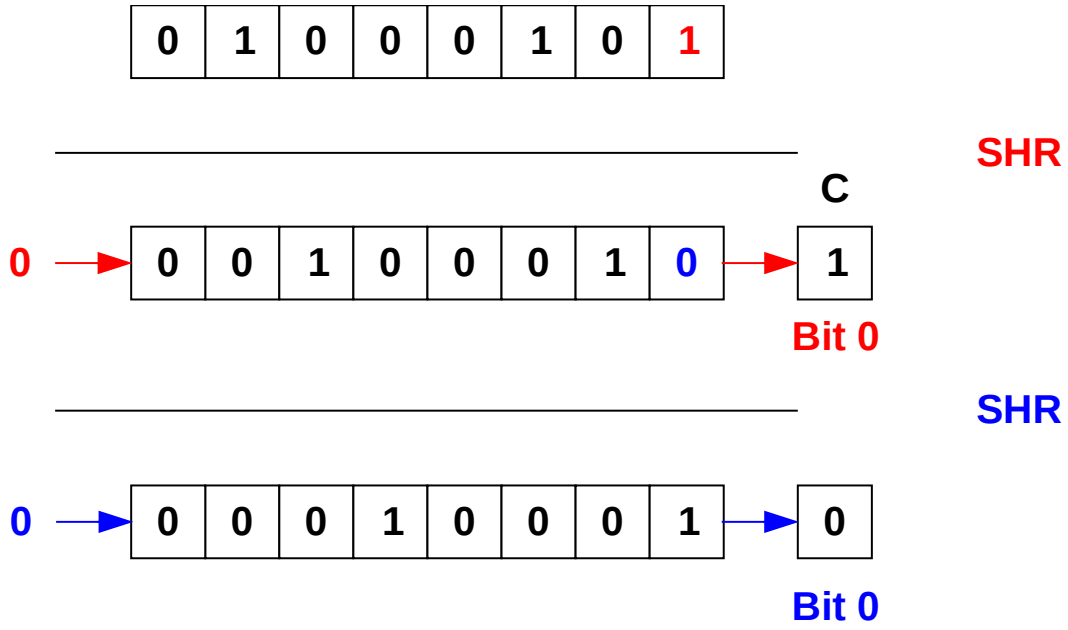
```
LDR     X,#H30
NEG     X
```



Exemplo – SHR

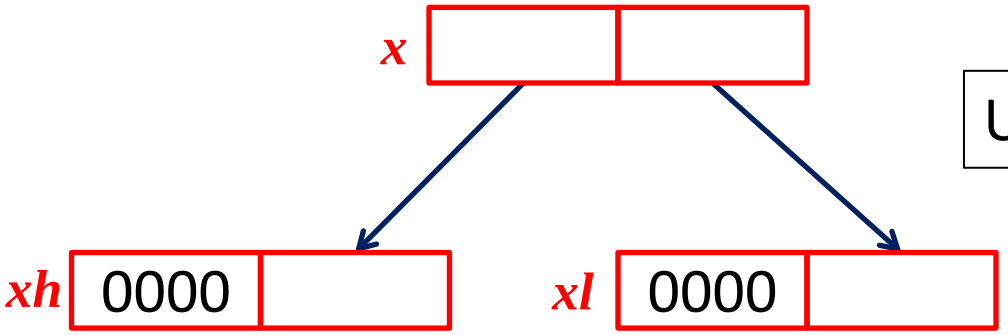
- Sequência de instruções

```
LDR     B,#H45
SHR     B
SHR     B
```



Exercício 9 (10 min)

- Implementar em *assembly* do RAMSES o algoritmo representado abaixo
  - Considere "***x***", "***xh***" e "***xl***" como "*unsigned char*"



Utilize SHR para deslocar bits

Desvios

- Relembrando: nos desvios, o último acesso à memória não é efetuado durante a execução da instrução
- Este último acesso é realizado no próximo ciclo de busca (*fetch*) de instrução
- Portanto, não faz sentido usar desvios com modo de endereçamento imediato

Desvios
(Iguais ao NEANDER)

- JMP <*end*>
- JN <*end*>
- JZ <*end*>

Desvios (Novidade)

- JC *<end>*

  if (C==1) PC ← n

RAMSES

Instrução	Operação	N	Z	C	Descrição
NOP	Nenhuma operação				Nenhuma operação
STR r end	MEM(end) ← r				Armazena registrador na memória (**store**)
LDR r end	r ← MEM(end)	↕	↕		Carrega registrador da memória (**load**)
ADD r end	r ← r + MEM(end)	↕	↕	↕	Adição (**soma** memória ao registrador)
OR r end	r ← r ∨ MEM(end)	↕	↕		"**ou**" lógico
AND r end	r ← r ∧ MEM(end)	↕	↕		"**and**" lógico
NOT r	r ← ¬ r	↕	↕		Inverte (**complementa** os bits do registrador)
SUB r end	r ← r - MEM(end)	↕	↕	↕(1)	Subtração (**subtrai** memória do registrador)
JMP end	PC ← end(2)				Desvio incondicional (**jump**)
JN end	if N=1 then PC ← end(2)				Desvio condicional se < 0 (**jump on negative**)
JZ end	if Z=1 then PC ← end(2)				Desvio condicional se =0 (**jump on zero**)
JC end	if C=1 then PC ← end(2)				Desvio condicional se carry=1 (**jump on carry**)
NEG r	r ← 0 – r	↕	↕	↕	Troca de sinal (**negate**)
SHR r	0 → [→] → C	↕	↕	↕	Deslocamento para direita (**shift right**)
HLT	parada				Parada (**halt**)

Exercício 10

- Implementar um programa capaz de calcular o **número de caracteres** de um vetor, conforme o programa abaixo
  - O resultado deverá ser colocado no endereço H80
  - O vetor inicia no endereço H81
- O "vetor" é um *string* em "C", que termina com '\0'

```
unsigned char n,i;
char vetor[50];

n = i = 0;
while ( vetor[i] != 0 ) {
    n++;
    i++;
}
```

Utilize os três registradores!

Utilize modo **indexado** para acesso ao vetor

(usar indexado sempre que o vetor estiver em endereço fixo)

Exercício 11

- Implementar um programa capaz de calcular o número de caracteres de um vetor, conforme o programa abaixo
- O “vetor” é um *string* em “C”, que termina com ‘\0’

```
unsigned char n;
char *p;
char vetor[50];

p = vetor;
n = 0;
while ( *p != 0 ) {
    ++n;
    ++p;
}
```

Utilize os três registradores!

Utilize modo **indireto** para acesso ao vetor

(usar indireto sempre que forem usados ponteiros)

Exercício 12

- Implementar o cálculo da raiz quadrada, conforme o seguinte algoritmo
  - N: Número que se deseja calcular a raiz
  - K: Valor da raiz
- O algoritmo implementado baseia-se na expressão

$$N = \sum_{k=1}^{\sqrt{N}} (2k - 1)$$

  - Conta-se quantas vezes é necessário somar (2K-1), para atingir (N)
    - Esse número de vezes (K) é a raiz inteira de (N)

```
/*
     K = SQRT(N)
*/
unsigned char N, I, S, K;
S = K = I = 1;
while (S < N) {
     I += 2;
     S += I;
     K++;
}
```

RAMSES
Instruções

Prof. Sérgio Luis Cechin

.inf INSTITUTO DE INFORMÁTICA UFRGS