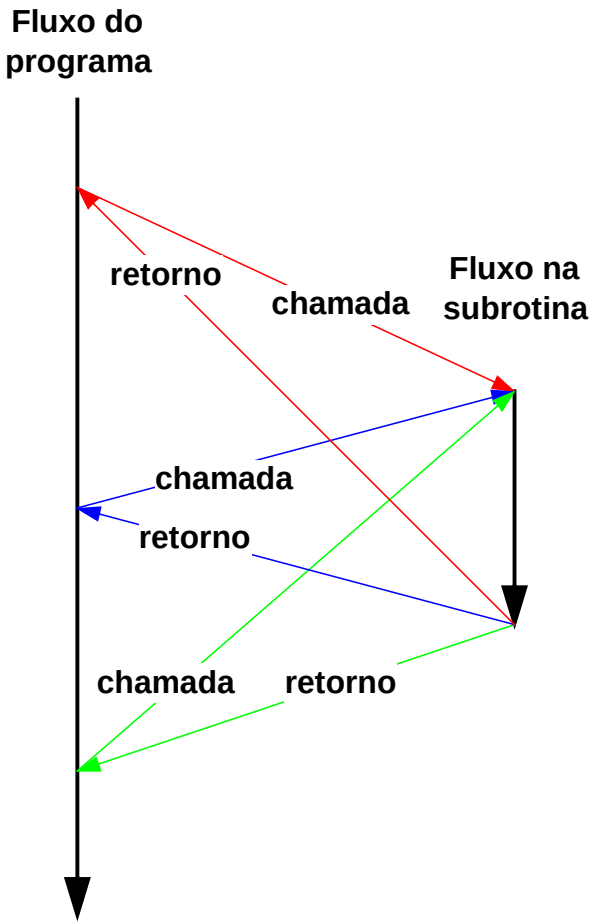
RAMSES
Subrotinas

Prof. Sérgio Luis Cechin

.inf
INSTITUTO
DE INFORMÁTICA
UFRGS

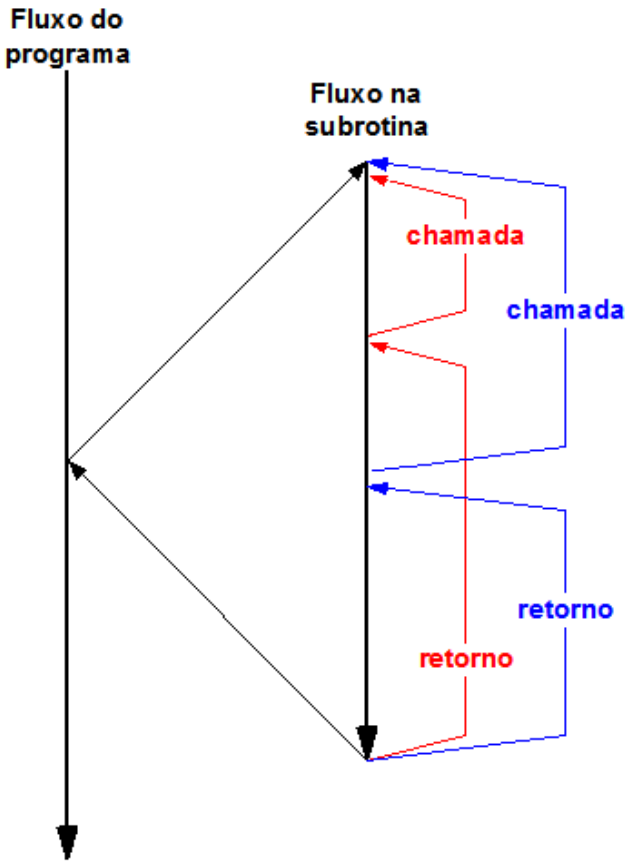
Princípio de Operação

- Sequência de instruções que podem ser ativadas de qualquer parte do programa
- Ao encerrarem, retornam ao ponto de ativação
- JSR = JMP + Armazenamento do endereço de retorno



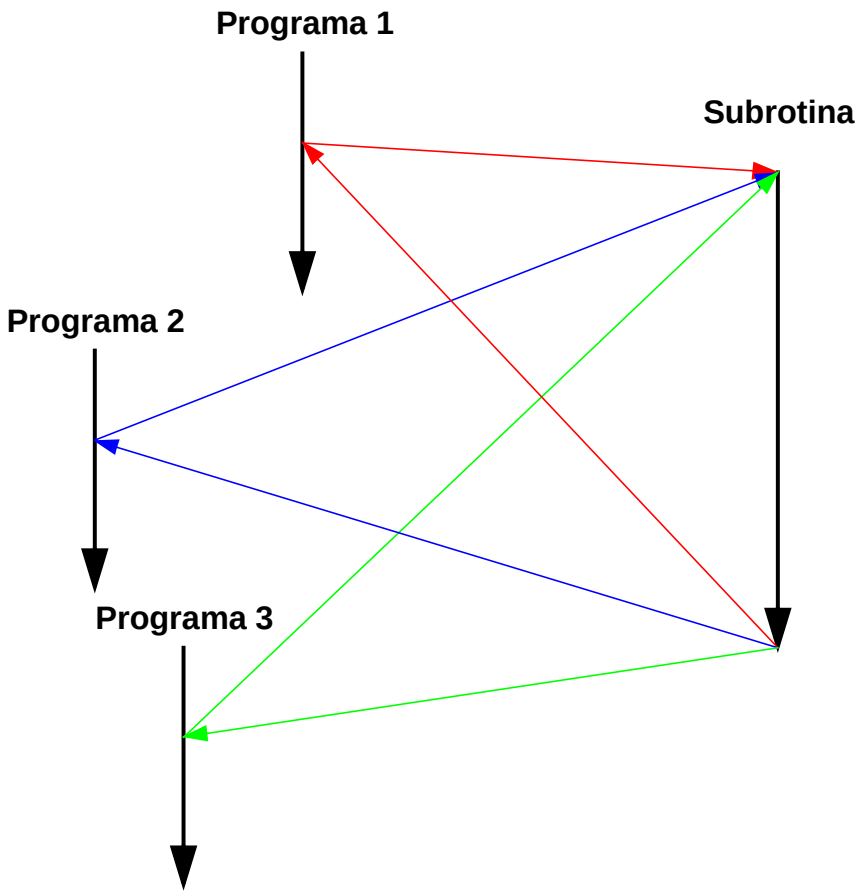
Recursividade

- São aquelas capazes de chamar a si próprias



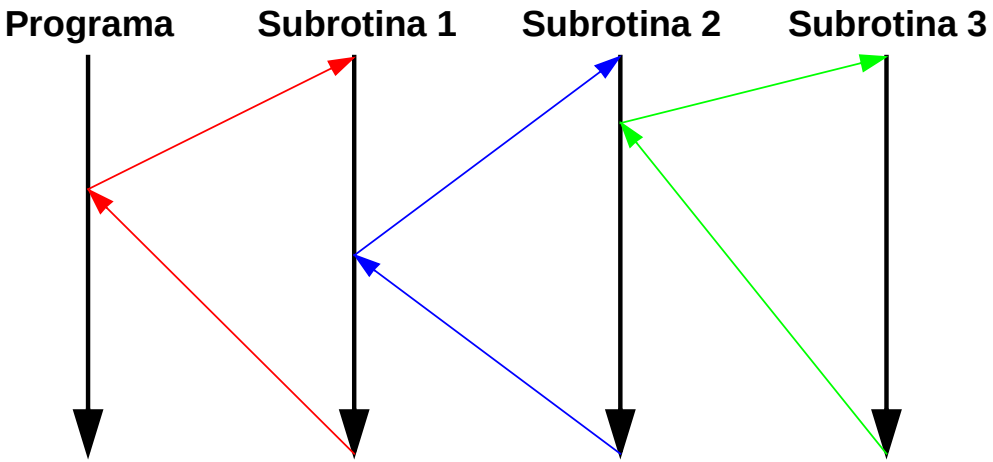
Reentrância

- Vários programas podem chamar a mesma rotina (o mesmo trecho de código)
- Chamadas simultaneas



Aninhamento

- Número de níveis de aninhamento
  - Possibilidade de uma subrotina chamar outra
  - “Subrotinas chamadas por subrotinas”



Desvios para subrotina
(no RAMSES)

- JSR
  - Desvio para subrotina:
    - PC $\rightarrow$ MEM(end)
    - PC $\leftarrow$ end+1
  - JSR end
  - Modos de endereçamento de “end”
    - Direto (end=n)
    - Indireto (end=n,I)
    - Indexado (end=n,X)
  - No final da subrotina:
    - Comando para retornar ao ponto de ativação da subrotina
    - JMP end,I

~~**JSR** #05~~
JSR H80
JSR H81,**I**
JSR H82,**X**

Exemplo

Endereço	Código		Mnemônico	
...			(programa)	
23	C0 60		JSR	Rotina
...			(programa)	
60	00	Rotina:	NOP	
...			(subrotina)	
70	81 60		JMP	Rotina,I

Entrada na subrotina

- ... imediatamente antes da chamada da subrotina
  - PC=23
- Chamada da subrotina
  - Fase de busca
    - RI ← MEM(PC), PC ← PC+1
    - RI=0C0H, PC=024H
  - Fase de execução
    - Leitura do segundo byte
      - End ← MEM(PC), PC ← PC+1
      - End=060H, PC=025H
    - Execução
      - MEM(End) ← PC, PC ← End+1
      - MEM(060H)=025H, PC=061H

Encerramento da subrotina

- Subrotina executada a partir de 61 até 6F
- Retorno: última instrução da subrotina (PC=70)
  - Fase de busca
    - RI ← MEM(PC), PC ← PC+1
    - RI=081H, PC=071H
  - Fase de execução
    - Leitura do segundo byte
      - End ← MEM(PC), PC ← PC+1
      - End=060H, PC=072H
    - Execução (JMP n,I)
      - PC ← MEM(End)
      - PC=025H

Características do JSR no RAMSES

- Recursividade
  - Não permitida
- Reentrância
  - Não permitida
- Níveis de aninhamento
  - Ilimitados

Passagem de parâmetros

1. Quanto ao tipo dos parâmetros
   - Por valor
   - Por referência
2. Quanto ao local de armazenamento
   - Em registro
   - Em memória
     - Em área específica de dados
     - Após a chamada
3. Quanto ao acesso
   - Acesso absoluto = endereços fixos
     - Modo direto
   - Acesso relativo = endereços relativos
     - Modos indexado ou indireto

Quanto ao tipo de parâmetro

Valor

X

Referência

Passagem por VALOR

- O parâmetro é o DADO propriamente dito
- Antes da chamada
  - Copia-se o VALOR original para a memória ou registrador
- Dentro da rotina
  - Utiliza-se a cópia da memória ou registrador
  - Modo DIRETO do Ramses
- Não permite a atualização do valor original
  - Só é capaz de alterar o parâmetro local

Exemplo – passagem por VALOR

```
1          ORG     128
2  Valor:  DAB     0, 0, 0, 0, 0   ; Variável do programa principal
3  param:  DB      0               ; Parâmetro local da rotina
4
5          ORG 0
6
7          ; Inc (Valor[3])
8          LDR     A, Valor+3      ; Cópia "Valor[3]" para "param"
9          STR     A, param
10         JSR     Inc
11         HLT
12
13 ; void Inc (unsigned char param)
14 Inc:    NOP
15         LDR     A, param        ; Operação sobre a cópia ("param")
16         ADD     A,#1
17         STR     A, param        ; Altera apenas o parâmetro local
18         JMP     Inc, I
```

Passagem por REFERÊNCIA

- O parâmetro é o PONTEIRO para o dado
- Antes da chamada
  - Copia-se o ENDEREÇO do valor original para a memória ou registrador
  - O ponteiro é passado por VALOR
- Dentro da rotina
  - Utiliza-se o acesso aos dados através do ponteiro
  - Modo INDIRETO, no RAMSES
- Pode-se alterar o valor original que foi passado
  - Alteração através do ponteiro
- Não permite a alteração do ponteiro original (se existir)

Exemplo – passagem por REF.

```
1          ORG     128
2  Valor:  DAB     0, 0, 0, 0, 0   ; Variável do programa principal
3  param:  DB      0               ; Parâmetro local da rotina
4
5          ORG 0
6
7          ; Inc (&Valor[3])
8          LDR     A, #Valor+3     ; Cópia "&Valor[3]" para "param"
9          STR     A, param
10         JSR     Inc
11         HLT
12
13 ; void Inc (unsigned char *param)
14 Inc:    NOP
15         LDR     A, param, I     ; Operação sobre a cópia ("param")
16         ADD     A,#1
17         STR     A, param, I     ; Altera o valor original
18         JMP     Inc, I
```

Quanto ao local de armazenamento dos parâmetros

Registrador

X

Memória

Passagem por REGISTRADOR

- O parâmetro está nos registradores do processador
- Antes da chamada
  - Copia-se o valor original em um registrador
  - Pode ser um VALOR ou um PONTEIRO
- Dentro do rotina
  - Se for um VALOR, usa-se diretamente
  - Se for um PONTEIRO, depende dos modos de endereçamento
- É muito eficiente

Exemplo – passagem por reg.

```
1          ORG     128
2  Valor:  DAB     0, 0, 0, 0, 0   ; Variável do programa principal
3
4          ORG     0
5
6  ; Inc (Valor[3])
7          LDR     B, Valor+3      ; Cópia "Valor[3]" para o reg B
8          JSR     Inc
9          HLT
10
11 ; void Inc (unsigned char param)
12 Inc:    NOP
13         ADD     B,#1            ; Operação sobre a cópia (reg B)
14         JMP     Inc, I
```

Passagem em Memória

- Em área específica
  - Ver exemplos “por valor” e “por referência”
- Em área após a chamada
  - A “área específica” está imediatamente após a chamada da rotina (ver próximo slide)
- É menos eficiente do que a passagem por registrador

Passagem em Memória Após a Chamada

- Os parâmetros estão nos bytes imediatamente posteriores à chamada da rotina
- Em geral, os parâmetros são "*read only*"
  - Mas, pode-se alterar esse parâmetros para uso do programa principal
- A rotina deve efetuar o ajuste do endereço de retorno

Exemplo – após a chamada

```
1          ORG     0
2
3          ; Soma(5,7);
4          JSR     Soma            ; Soma os dois bytes que seguem
5          DB      5
6          DB      7
7
8          HLT
9
10 ; void Soma(unsigned char p1, unsigned char p2)
11 Soma:   NOP
12         LDR     A, Soma, I      ; Pega parâmetro "p1"
13
14         LDR     B,Soma          ; Inc. ponteiro para o parâmetro "p2"
15         ADD     B,#1
16         STR     B,Soma
17
18         ADD     A, Soma, I      ; Soma com o parâmetro "p2"
19
20         LDR     B, Soma         ; Ajusta endereço de retorno
21         ADD     B, #1
22         STR     B, Soma
23
24         JMP     Soma, I
```

Quanto ao acesso

Absoluto

X

Relativo

Acesso Absoluto

- A referência (ponteiro) indica a posição de memória do parâmetro
- Ver variável “param” no exemplo a seguir

Ex: Acesso Absoluto

```
1
2           ORG     128
3   vetor:  DAB     1, 2, 3, 4, 5, 6, 7, 8, 9, 0
4   param:  DB      0
5
6           ORG     0
7
8           ; n = vetor[0] + vetor[1]
9           LDR     A,#vetor
10          STR     A,param
11          JSR     SAbs
12
13          ; n = vetor[5] + vetor[6]
14          LDR     A,#vetor+5
15          STR     A,param
16          JSR     SAbs
17
18          HLT
```

```
20  SAbs:   NOP
21          LDR     A,param,I
22
23          LDR     B,param
24          ADD     A,#1
25          STR     A,param
26
27          ADD     A,param,I
28          JMP     SAbs,I
29
```

O valor a ser processado é passado em endereço absoluto (em “param”)

Acesso Relativo

- A referência (ponteiro) indica a posição do parâmetros, a partir de uma posição de referência
- Ver valor passado no registrador “X”, no exemplo a seguir

Ex: Acesso Relativo

```
40          ORG     128
41  vetor:  DAB     1, 2, 3, 4, 5, 6, 7, 8, 9, 0
42  param:  DB      0
43
44          ORG     0
45
46          ; n = vetor[0] + vetor[1]
47          LDR     X,#0
48          JSR     SRel
49
50          ; n = vetor[5] + vetor[6]
51          LDR     X,#5
52          JSR     SRel
53
54          HLT
```

```
56  SRel:   NOP
57          LDR     A,vetor,X
58          ADD     X,#1
59          ADD     A,vetor,X
60          JMP     SAbs,I
```

```
62  SRel2:  NOP
63          LDR     A,vetor,X
64          ADD     A,vetor+1,X
65          JMP     SAbs,I
```

O valor a ser processado é passado de forma relativa

“vetor” é a referência

“X” é a posição relativa, em relação à referência

RAMSES
Subrotinas

Prof. Sérgio Luis Cechin

.inf
INSTITUTO
DE INFORMÁTICA
UFRGS