Conjunto de Instruções Thumb2 (continuação)

Prof. Hugo Vieira Neto

2020/2

Sub-rotinas

- Há um tipo especial de instrução de **desvio com ligação** que permite posterior retorno à sequência original do fluxo do programa
- Esse tipo de instrução de **desvio com ligação** é usada com sub-rotinas
  - Funcionam como uma espécie de superinstrução, montada a partir das instruções do processador
  - Analogia: uma instrução está para uma sub-rotina assim como um tijolo está para uma parede

Sub-rotinas em Assembly

- Observar dois conceitos principais:
  1. Uma sub-rotina deve poder ser chamada de qualquer parte do programa
  2. Uma vez terminada, a sub-rotina deve retornar à mesma parte do programa de onde foi chamada
- Uma sub-rotina que chama outra é conhecida como “chamadora” (caller)
- Uma sub-rotina que é chamada por outra é conhecida como “chamada” (callee)

Desvio com Ligação

- Instrução Branch with Link (direto)
  - Mnemônico: BL
  - Formato: BL <label>
  - Operação: LR := PC (próx. instr.), PC := label.
- O registrador LR (link register) armazena o endereço de retorno da sub-rotina
- Diz-se que a sub-rotina é “chamada” com a instrução BL e que “retorna” com a BX LR

Componentes de Sub-rotinas

- Ponto de entrada
  - Endereço da primeira instrução da sub-rotina
- Parâmetros de entrada
  - Registradores ou endereços da memória que contêm os parâmetros de entrada da sub-rotina
- Parâmetros de retorno
  - Registradores ou endereços da memória que contêm os parâmetros de retorno da sub-rotina
- Memória de trabalho
  - Registradores ou endereços da memória auxiliares para a sub-rotina poder executar sua função

Exemplo de Sub-rotina

- Sub-rotina para adição de quatro operandos
  - Entrada: registradores R0 a R3
  - Saída: registrador R0

```
Add4op:   ADD R0, R0, R1   ; adição...
          ADD R0, R0, R2   ; acumulada...
          ADD R0, R0, R3   ; em R0
          BX LR            ; retorno
```

Exemplo de Uso (Add4op)

```
main:   MOV R0, #11
        MOV R1, #22
        MOV R2, #33
        MOV R3, #44
        BL App4op ; R0 := R0 + R1 + R2 + R3
        MOV R1, #55
        BL App4op ; R0 := R0 + R1 + R2 + R3
loop:   B loop
```

Exercício 6

- Transformar os programas dos Exercícios 4 e 5 em sub-rotinas para multiplicação e divisão inteiras de valores de 8 bits
- Sub-rotina Mul8b:
  - Entrada: registradores R0 e R1
  - Saída: registrador R2 (produto)
- Sub-rotina Div8b:
  - Entrada: registradores R0 e R1
  - Saída: registradores R2 (quociente) e R3 (resto)

Exercício 6

- Utilizar as duas sub-rotinas (Mul8b e Div8b) em um programa que efetue a multiplicação e a divisão inteiras de valores armazenados nos registradores R8 e R9, respectivamente
- Ao final do programa, o produto deve estar armazenado no registrador R10, o quociente no registrador R11 e o resto no registrador R12

Exercício 6

- Simular o programa resultante no depurador, executando-o passo-a-passo e observando os resultados parciais de cada instrução
- Prestar atenção particularmente ao que acontece com os registradores PC e LR ao executar as instruções de chamada (BL) e de retorno (BX LR) de sub-rotina

Conceito de Pilha

- A pilha é uma estrutura para armazenamento temporário de dados na memória
- Utiliza o registrador SP como ponteiro para o topo da pilha
- A instrução PUSH coloca dados (conteúdo de registradores) no topo da pilha
- A instrução POP retira dados do topo da pilha, carregando-os em registradores

Conceito de Pilha

- É uma estrutura de armazenamento tipo LIFO (Last In, First Out)
  - O último dado colocado (topo da pilha) será o primeiro a ser retirado

Uso da Pilha

- O registrador SP é inicializado no reset do processador ARM Cortex-M4 e deve apontar para uma região da memória de dados
- O espaço de memória de dados reservado para a pilha (stack) é definido na configuração do ligador (linker)
- O uso das instruções PUSH e POP deve ser balanceado para evitar overflow/underflow

Uso da Pilha

- A pilha é utilizada para:
  - Passagem de parâmetros para sub-rotinas
  - Armazenamento temporário do conteúdo de registradores em sub-rotinas
  - Armazenamento temporário de endereços de retorno de sub-rotinas aninhadas
  - Armazenamento de variáveis locais (linguagem C)

Uso da Pilha

- Instrução Push
  - Mnemônico: PUSH
  - Formato: PUSH <reglist>
  - Operação: salva lista de registradores em [SP].
- Instrução Pop
  - Mnemônico: POP
  - Formato: POP <reglist>
  - Operação: recupera lista de registradores de [SP].

Uso da Pilha

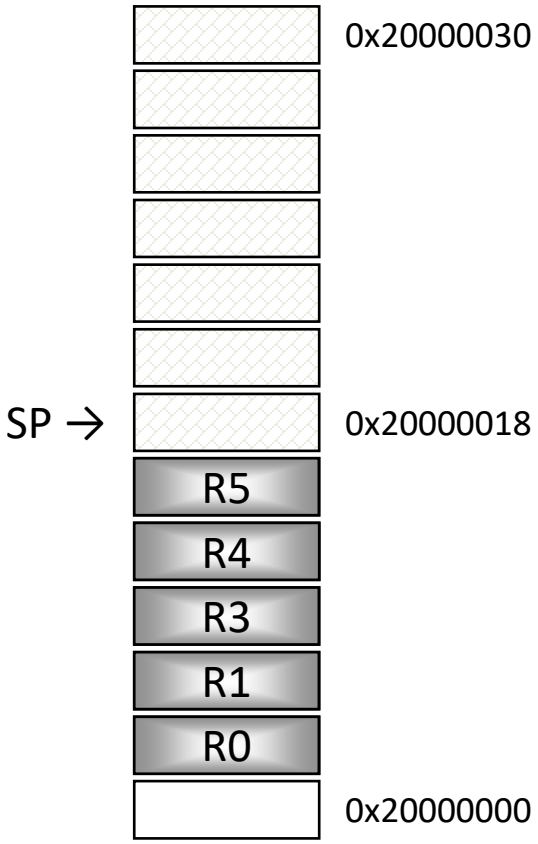
- A pilha do Cortex-M4 é do tipo descendente cheia (full descending stack)
  - Descendente: SP é decrementado no PUSH
  - Cheia: SP aponta sempre para um dado válido
- Ordem de execução:
  - PUSH: primeiro decrementa SP, depois coloca dado no topo da pilha
  - POP: primeiro retira o dado do topo da pilha, depois incrementa SP

Exemplos de Uso da Pilha

- PUSH {R0}
  - Ações: SP--, [SP] := R0
- POP {R0}
  - Ações: R0 := [SP], SP++
- PUSH {R1,R2}
  - Ações: SP--, [SP] := R2, SP--, [SP] := R1
- POP {R1,R2}
  - Ações: R1 := [SP], SP++, R2 := [SP], SP++

Exemplos de Uso da Pilha

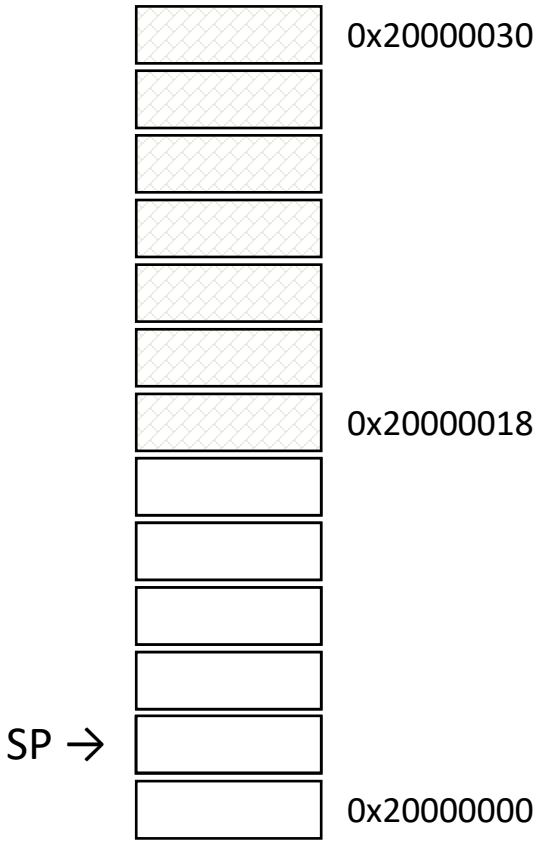
PUSH {R0,R1,R3-R5}



Exemplos de Uso da Pilha

PUSH {R0,R1,R3-R5}

POP {R0,R1,R3-R5}



Exercício 7

- Simule o programa a seguir passo-a-passo no depurador
- Observe os registradores, particularmente o comportamento do SP (janela Registers)
- Observe o conteúdo da pilha (janela Stack)
- Faça experimentos trocando a ordem dos registradores nas instruções PUSH e POP e observe o que acontece

Exercício 7

```
main:   MOV R0, #0x11
        MOV R1, #0x22
        MOV R2, #0x33
        PUSH {R0, R1, R2}
        POP {R3, R4, R5}
        EOR R0, R0, #0xF0 ; ou exclusivo
        EOR R1, R1, #0x0F
        EOR R2, R2, #0xFF
        B main
```

Particularidades do Cortex-M

Processador ARM Cortex-M4

1. PUSH e POP de múltiplos registradores por instrução
2. Salvamento de endereços de retorno de sub-rotinas no registrador LR
   - Execução mais eficiente
   - Sub-rotinas aninhadas: tratamento manual

   PUSH LR / BL sub / POP LR

Processadores Convencionais

1. PUSH e POP de um único registrador por instrução
2. Salvamento de endereços de retorno de sub-rotinas na pilha
   - Execução menos eficiente
   - Sub-rotinas aninhadas: tratamento automático

   CALL sub (ex: 8051)

Padrão AAPCS para Sub-rotinas

- *ARM Architecture Procedure Call Standard*
- Passagem de parâmetros para a sub-rotina:
  - Primeiros parâmetros em R0, R1, R2 e R3
  - Demais parâmetros pela pilha
- Retorno da sub-rotina:
  - R0 (32 bits)
  - R1:R0 (64 bits)
- Alinhamento da pilha deve ser de 64 bits:
  - PUSH/POP sempre de quantidade par de registradores

Padrão AAPCS para Sub-rotinas

- Regras de “posse” dos registradores
  - R0 a R3 e R12 são da sub-rotina chamada (a sub-rotina chamadora deve salvá-los na pilha se for necessário preservá-los)
  - R4 a R11 são da sub-rotina chamadora (a sub-rotina chamada deve salvá-los na pilha se for necessário preservá-los)
- O padrão AAPCS permite compatibilidade entre linguagens C e Assembly