

# Linguagens de montagem

## Capítulo 13 - ARM - Entrada e Saída

Ricardo Anido  
Instituto de Computação  
Unicamp

- ▶ No processador ARM a entrada e saída é mapeada na memória, ou seja, o ARM não possui instruções específicas para acessar entrada e saída.
- ▶ Para comunicação do processador com os dispositivos são usadas as instruções normais de acesso à memória (LDR e STR).
- ▶ Igum intervalo (ou intervalos) de endereços de memória deve ser reservado para dispositivos de entrada e saída.
- ▶ Geralmente os endereços reservados para E/S no ARM são os do final do espaço de endereçamento (endereços de valores altos), mas isso depende da arquitetura de hardware do sistema.

Considerando o mostrador de sete segmentos e o teclado simples, escreva um programa em linguagem de montagem do ARM para atualizar continuamente o mostrador com o valor da tecla pressionada. Considere que o endereço da porta de dados do mostrador seja 0x90000, e os endereços das portas de estado e de dados do teclado sejam respectivamente 0x90010 e 0x90011.

@ programa para atualizar constantemente o mostrador com o valor  
@ de uma tecla pressionada

@ endereços dispositivos

.equ DISPLAY, 0x90000

.equ KBD\_DATA, 0x90010

.equ KBD\_STATUS, 0x90011

@ constantes

.equ KBD\_READY, 0x1

.equ KBD\_OVRN, 0x2

.org 0x100

le\_tecla:

```
    ldr    r1,=KBD_STATUS    @ porta de estado do teclado
```

le\_tecla1:

```
    ldr    r2,[r1]           @ lê estado do teclado
```

```
    tst    r2,#KBD_READY     @ verifica se tecla foi pressionada
```

```
    beq    le_tecla1         @ se não, continua no laço
```

```
    ldr    r1,=KBD_DATA      @ r1 tem porta de dados do teclado
```

```
    ldr    r3,[r1]           @ r3 tem valor da tecla pressionada
```

```
    ldr    r2,=DISPLAY       @ r2 tem porta do mostrador
```

```
    ldr    r4,=digitos       @ tabela com padrões de bits
```

```
    ldrb   r0,[r4,r3]        @ padrão de bits para valor da tecla
```

```
    strb   r0,[r2]           @ seta valor no mostrador
```

```
    b      le_tecla          @ e continua para sempre
```

digitos:

```
.byte 0x7e,0x30,0x6d,0x79,0x33,0x5b,0x5f,0x70,0x7f,0x7b,0x4f,0x4f
```

# Interrupções e exceções

- ▶ Interrupções e exceções no ARM são associadas aos modos de operação do processador.
- ▶ O ARM possui duas interrupções externas: FIQ (interrupção rápida, do inglês *Fast Interrupt Request*) e IRQ (interrupção normal, do inglês *Interrupt Request*), e quatro tipos de exceções: *Reset*, *Data abort*, *Pre-fetch abort* e *Undefined instruction*.

# Interrupções e exceções

- ▶ As diferentes interrupções e exceções têm prioridades, indicadas por números de 1 a 6 (quanto menor o número maior a prioridade).
- ▶ Se um tipo mais prioritário de interrupção ou exceção está sendo atendido, os eventos de outros tipos ficam pendentes.
- ▶ Para as interrupções externas FIQ e IRQ o registrador CPSR mantém dois bits de controle, respectivamente bits F e I, que indicam se a interrupção deve ser tratada. Se o bit de controle correspondente no registrador CPSR é igual a zero, a interrupção está habilitada e é atendida; caso o bit correspondente seja igual a um (interrupção desabilitada), a interrupção fica pendente.

# Interrupções e exceções

- ▶ As diferentes interrupções e exceções têm prioridades, indicadas por números de 1 a 6 (quanto menor o número maior a prioridade).
- ▶ Se um tipo mais prioritário de interrupção ou exceção está sendo atendido, os eventos de outros tipos ficam pendentes.
- ▶ Para as interrupções externas FIQ e IRQ o registrador CPSR mantém dois bits de controle, respectivamente bits F e I, que indicam se a interrupção deve ser tratada. Se o bit de controle correspondente no registrador CPSR é igual a zero, a interrupção está habilitada e é atendida; caso o bit correspondente seja igual a um (interrupção desabilitada), a interrupção fica pendente.



# Interrupções e exceções

Interrupção/Exceção	Prioridade	Número
<i>Reset</i>	1	0
<i>Data Abort</i>	2	4
<i>FIQ</i>	3	7
<i>IRQ</i>	4	6
<i>Prefetch Abort</i>	5	3
<i>SVC</i>	6	2
<i>Undefined Instruction</i>	6	1

# Mecanismo de interrupção

- ▶ muda o modo de operação para o modo correspondente ao evento;
- ▶ copia o registrador CPSR no registrador SPSR do modo correspondente;
- ▶ armazena o endereço de retorno da interrupção no registrador `lr (r14)` do modo correspondente;
- ▶ desvia para o endereço *número\_da\_interrupção*  $\times 4$  (elemento do vetor de interrupção correspondente ao tipo da interrupção ou exceção, Figura ??).

# Vetor de interrupções

Normalmente, cada entrada no vetor de interrupções contém uma instrução de desvio incondicional para a rotina de interrupção correspondente.

```
.org 0
vetor_int:
    b      trata_reset           @ tipo 0, Reset
    ldr    pc,=trata_undef       @ tipo 1, Undefined Instruction
    b      trata_svc            @ tipo 2, SVC
    ldr    pc,=trata_prefetch    @ tipo 3, Prefetch abort
    ldr    pc,=trata_abort       @ tipo 4, Data abort
    .skip  4                     @ tipo 5, reservado
    ldr    pc,=trata_irq         @ tipo 6, IRQ
trata_fiq:                        @ tipo 7, FIQ
    ...                          @ tratador FIQ pode ser colocado aqui
```

# Retorno de interrupção

Para retornar do tratamento de uma interrupção ou exceção deve ser utilizada a instrução especial MOVS:

```
movs    pc,lr
```

que restaura o registrador de estado, copiando o registrador de estado SPSR para o registrador de estado CPSR do modo em que o processador estava quando a interrupção foi aceita, e desvia para o endereço de retorno armazenado no registrador `lr`, voltando a executar o código interrompido.