Linguagens de montagem
Capítulo 13 - ARM - Entrada e Saída

Ricardo Anido
Instituto de Computação
Unicamp

Entrada e saída

- 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).
- Algum 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.

Problema

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`.

Solução

```
@ 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
```

Solução

```
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
```

Exceções

- existe outro tipo de evento, interno ao processador, que também pode ser tratado com o mecanismo de interrupção: *Exceção*,
- Exceção ocorre devido a algum problema na execução de uma instrução.
- Exemplo: divisão por zero.
- Outro exemplo: *instrução inválida*, disparada quando o processador detecta que o código da instrução a ser executada não corresponde a uma instrução válida.

Exceções

- Alguns tipos de interrupção são pré-definidos como sendo reservados para exceções,
- O hardware, no caso de ocorrência de uma exceção, gera uma interrupção do tipo correspondente.

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
Reset	1	0
Data Abort	2	4
FIQ	3	7
IRQ	4	6
Prefetch Abort	5	3
SVC	6	2
Undefined Instruction	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.

Mudança de modo de execução

- O processador, ao iniciar, está no modo de execução Supervisor.
- Para permitir que o programador prepare cada modo de execução (Supervisor, IRQ, FIQ, etc) que pretende utilizar (preparar a pilha para o modo, por exemplo), o ARM inclui uma instrução que permite a mudança de um modo de execução para outro: MSR (move para registrador de estado).
- O modo de execução corrente é armazenado nos bits 4 a 0 do CPSR; assim, alterando o valor do CPSR é possível alterar o modo de execução corrente.
- Note ainda que a instrução MSR não pode ser executada no modo Usuário.

Mudança de modo de execução

Modos de operação definidos pelos bits de controle M[4:0] do CPSR

M[4:0]	Modo de Operação
10000	*User*
10001	*FIQ*
10010	*IRQ*
10011	*Supervisor*
10111	*Abort*
11011	*Undefined*
11111	*System*