QXD0133 - Arquitetura e Organização de Computadores II

Universidade Federal do Ceará - Campus Quixadá

Thiago Werlley
thiagowerlley@ufc.br

18 de outubro de 2025

Capítulo 9

Capítulo 9

Tratamento de exceções

Introdução

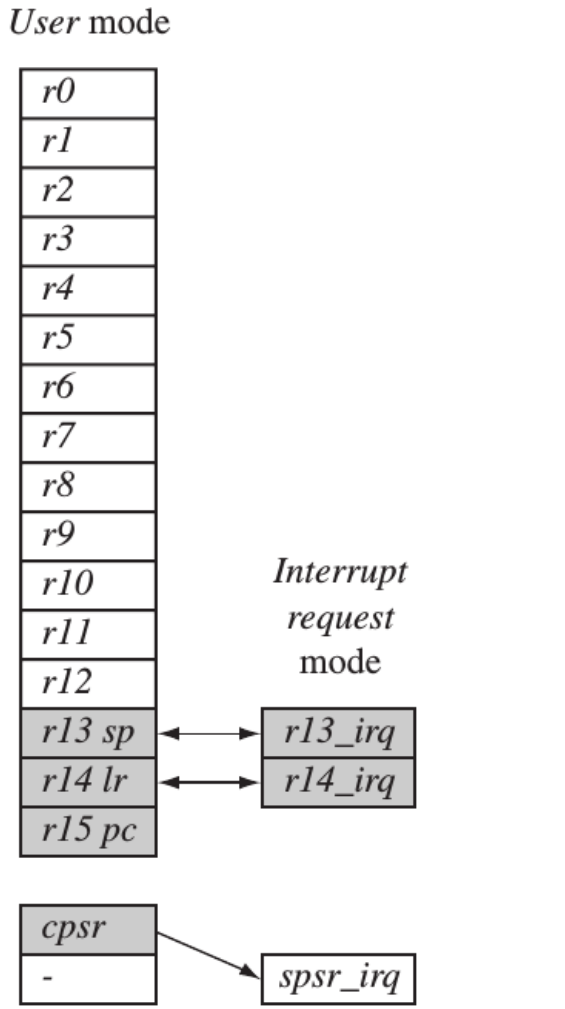
- O processador ARM possui sete exceções que podem interromper a execução sequencial normal de instruções:
  - Interrupção de dados;
  - Solicitação de interrupção rápida;
  - Solicitação de interrupção;
  - Interrupção de pré-busca;
  - Interrupção de software;
  - Redefinição;
  - Instruções indefinidas.

Tratamento de Exceção

- Exceptions:
  - Qualquer condição que interrompe a execução sequencial normal de instruções.
    - Ex. Erros de acesso, interrupções, etc.
- O tratamento de exceções é o método de processamento dessas exceções.
- A maioria das exceções possui um tratamento de exceção de software associado
- Exception Handler:
  - Rotina de software que é executada quando a exceção ocorre.

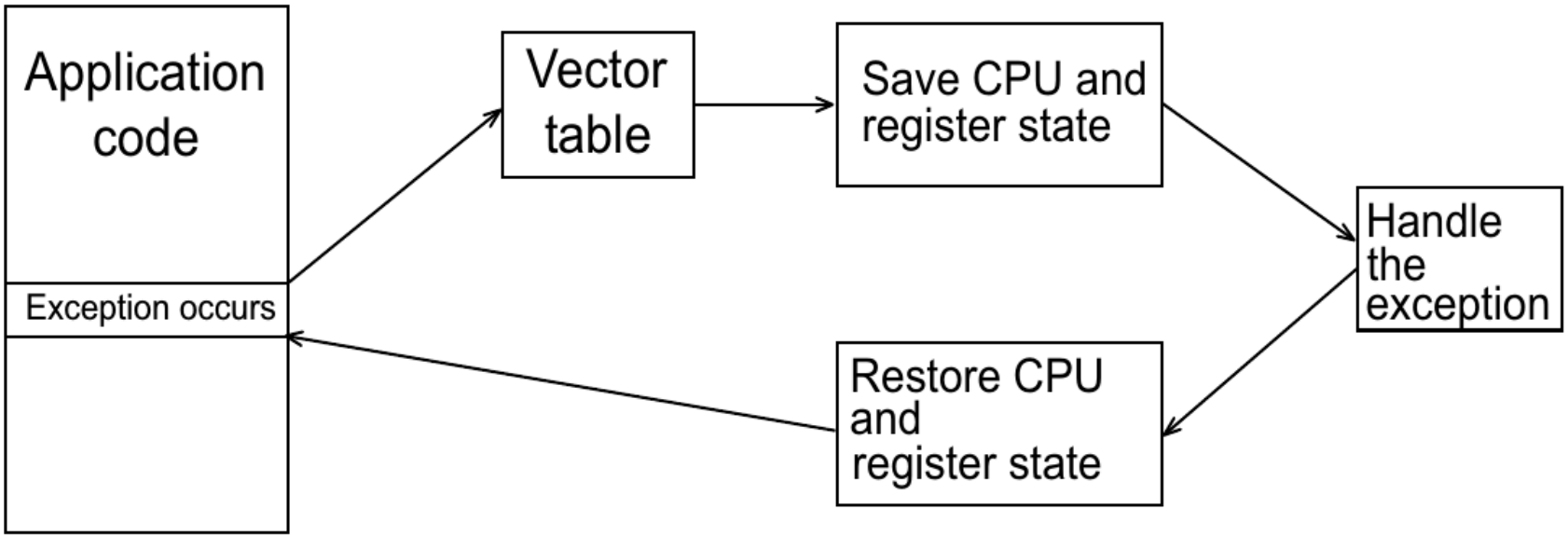
Exceções e modos do processador ARM

- Quando ocorre uma exceção que provoca a mudança de modo o processador:
  - Copia o **cpsr** para o **spsr** do modo correspondente
  - Muda para o estado ARM
  - Salva o endereço de retorno no **lr** do modo correspondente
  - Configura o **cpsr** para o modo de exceção
  - Desabilita as interrupções (dependendo da exceção)
  - Carrega o **pc** com o endereço da rotina de tratamento

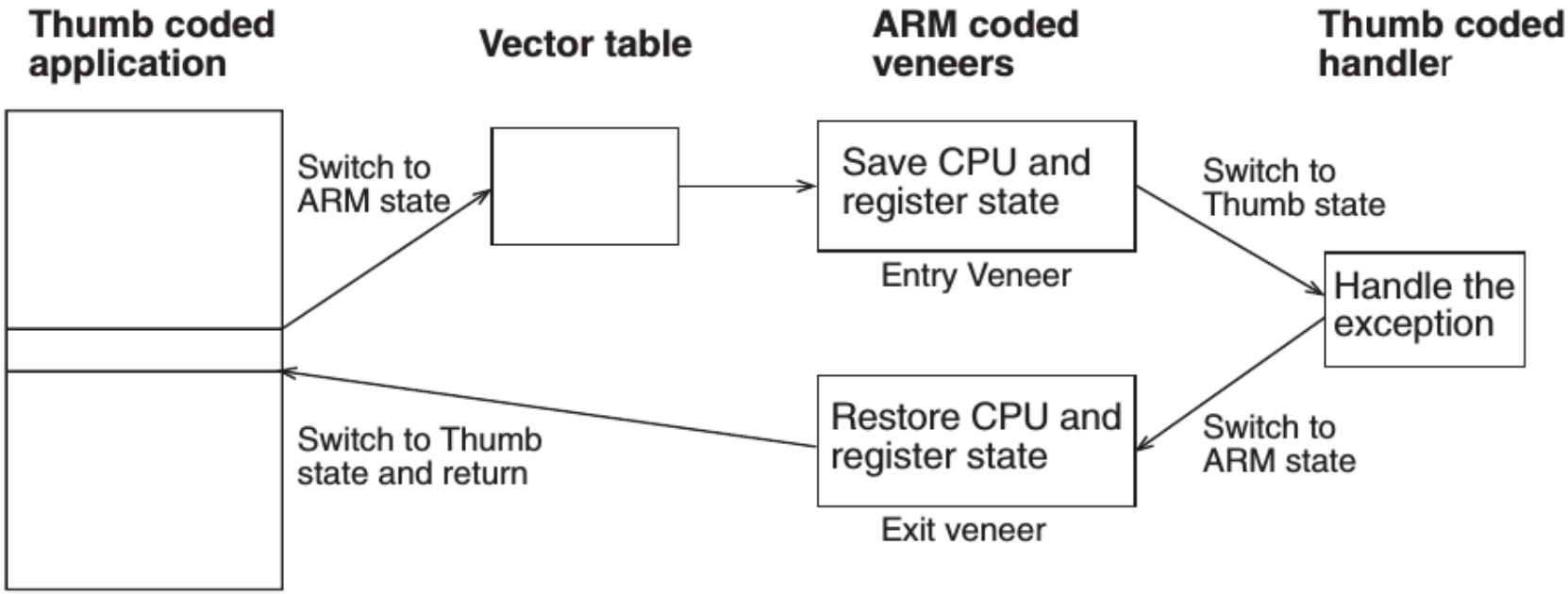


Changing mode on an exception.

Tratamento de exceções – ARM State



Tratamento de exceções – THUMB State



Exceções e modos

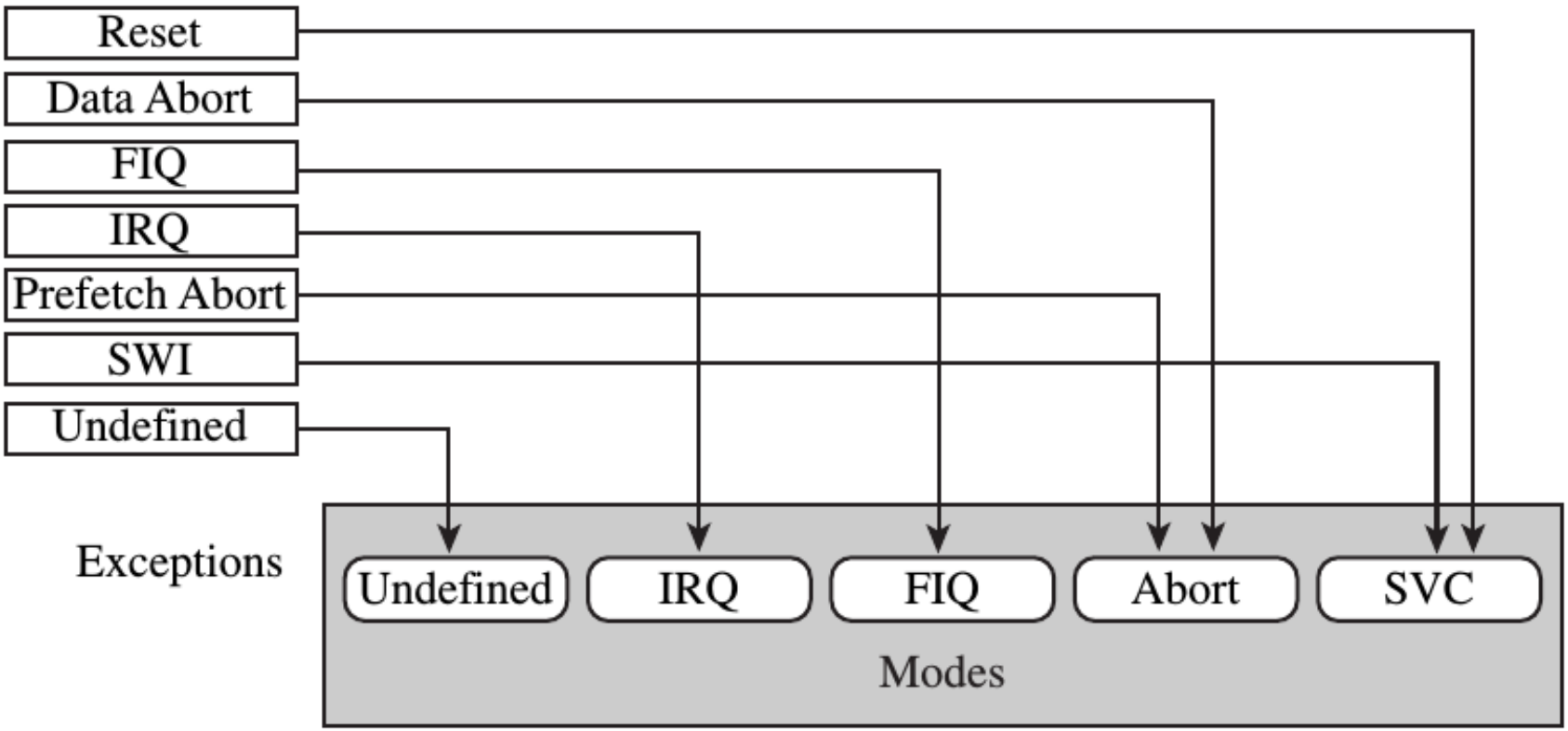


Tabela de Vetores

- Tabela de vetores é uma tabela de endereços aos quais o núcleo do ARM se ramifica quando uma exceção é gerada.
- Esses endereços geralmente contêm instruções de ramificação de um dos seguintes formulários:
  - `B <endereço>` - fornece uma ramificação relativa do PC.
  - `LDR pc, [pc, #offset]` - carrega o endereço do manipulador da memória para o PC.
  - `LDR pc, [pc,#-0xff0]` - carrega um endereço de rotina de serviço de interrupção específico do endereço 0xffffff030 para o PC.
  - `MOV pc, #imediato` - copia um valor imediato para o PC.

Tabela de vetores

Vector table and processor modes.

Exception	Mode	Vector table offset
Reset	SVC	+0x00
Undefined Instruction	UND	+0x04
Software Interrupt (SWI)	SVC	+0x08
Prefetch Abort	ABT	+0x0c
Data Abort	ABT	+0x10
Not assigned	—	+0x14
IRQ	IRQ	+0x18
FIQ	FIQ	+0x1c

Exception Handler (range de até 32MB)

```
Vector_Init_Block
                b       Reset_Addr
                b       Undefined_Addr
                b       SWI_Addr
                b       Prefetch_Addr
                b       Abort_Addr
                NOP                     ;Reserved vector
                b       IRQ_Addr
                b       FIQ_Addr

Reset_Addr      …
Undefined_Addr  …
SWI_Addr        …
Prefetch_Addr   …
Abort_Addr      …
IRQ_Addr        …
FIQ_Addr        …
```

Exception Handler

```
Vector_Init_Block
                LDR     PC, Reset_Addr
                LDR     PC, Undefined_Addr
                LDR     PC, SWI_Addr
                LDR     PC, Prefetch_Addr
                LDR     PC, Abort_Addr
                NOP                             ;Reserved vector
                LDR     PC, IRQ_Addr
                LDR     PC, FIQ_Addr

Reset_Addr      DCD     Start_Boot
Undefined_Addr  DCD     Undefined_Handler
SWI_Addr        DCD     SWI_Handler
Prefetch_Addr   DCD     Prefetch_Handler
Abort_Addr      DCD     Abort_Handler
                DCD     0                       ;Reserved vector
IRQ_Addr        DCD     IRQ_Handler
FIQ_Addr        DCD     FIQ_Handler
```

Exemplo de Tabela de vetores

```
0x00000000: 0xe59ffa38  RESET: > ldr  pc, [pc, #reset]
0x00000004: 0xea000502  UNDEF:   b    undInstr
0x00000008: 0xe59ffa38  SWI   :  ldr  pc, [pc, #swi]
0x0000000c: 0xe59ffa38  PABT  :  ldr  pc, [pc, #prefetch]
0x00000010: 0xe59ffa38  DABT  :  ldr  pc, [pc, #data]
0x00000014: 0xe59ffa38   -    :  ldr  pc, [pc, #notassigned]
0x00000018: 0xe59ffa38  IRQ   :  ldr  pc, [pc, #irq]
0x0000001c: 0xe59ffa38  FIQ   :  ldr  pc, [pc, #fiq]
```

Example vector table.

Prioridades das exceções

- Exceções podem ocorrer simultaneamente, portanto, o processador precisa adotar um mecanismo de prioridade.

Exception priority levels.

Exceptions	Priority	*I* bit	*F* bit
Reset	1	1	1
Data Abort	2	1	—
Fast Interrupt Request	3	1	1
Interrupt Request	4	1	—
Prefetch Abort	5	1	—
Software Interrupt	6	1	—
Undefined Instruction	6	1	—

Link Register Offsets

- Quando ocorre uma exceção, o registro do link é definido para um endereço específico com base no pc atual.
- Por exemplo, quando uma exceção IRQ é gerada, o registrador lr aponta para a última instrução executada mais 8.
- É necessário tomar cuidado para garantir que o manipulador de exceções não corrompa lr, pois ele é usado para retornar de um manipulador de exceções.
- A exceção IRQ é tomada somente após a execução da instrução atual, portanto o endereço de retorno deve apontar para a próxima instrução ou $lr - 4$.

Link Register Offsets

Exception	Address	Use
Reset	—	*lr* is not defined on a Reset
Data Abort	$lr - 8$	points to the instruction that caused the Data Abort exception
FIQ	$lr - 4$	return address from the FIQ handler
IRQ	$lr - 4$	return address from the IRQ handler
Prefetch Abort	$lr - 4$	points to the instruction that caused the Prefetch Abort exception
SWI	*lr*	points to the next instruction after the SWI instruction
Undefined Instruction	*lr*	points to the next instruction after the undefined instruction

- Exemplos:

```
handler
        <handler code>
        ...
        SUBS    pc, r14, #4
```

```
handler
        SUB     r14, r14, #4
        ...
        <handler code>
        ...
        MOVS    pc, r14
```

```
handler
        SUB     r14, r14, #4
        STMFD   r13!,{r0-r3, r14}
        ...
        <handler code>
        ...
        LDMFD   r13!,{r0-r3, pc}^
```

Exemplos

Exemplo que mostra um método típico de retornar de um handler de IRQ e FIQ usando a instrução SUBS

```
handler
<handler code>
...
SUBS pc, r14, #4 ; pc=r14-4
```

Como existe um S no final da instrução SUB e o pc é o registro de destino, o cpsr é restaurado automaticamente a partir do registro spsr.

Atribuindo Interrupções

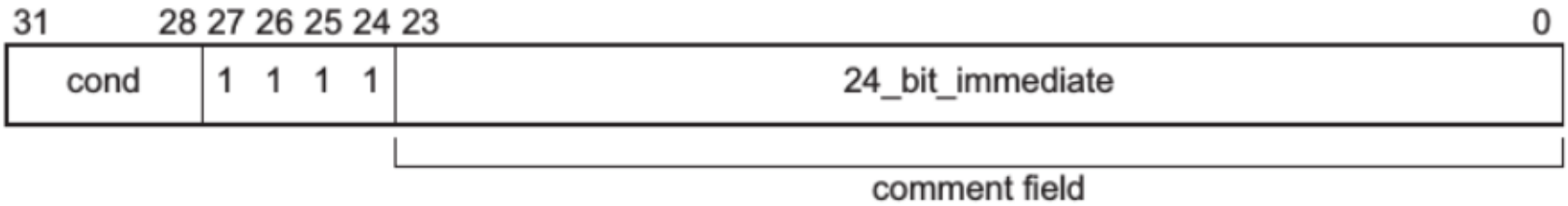
- Um **projetista de sistemas** pode decidir qual periférico de hardware pode produzir qual solicitação de interrupção.
- Essa decisão pode ser implementada em **hardware** ou **software** (ou ambos) e depende do sistema embarcado que está sendo usado.
- Quando se trata de atribuir interrupções, os projetistas de sistemas adotaram uma prática padrão de design:
  - **Interrupções de software** - são normalmente reservadas para chamar rotinas privilegiadas do sistema operacional.
  - **Solicitações de interrupção** - são normalmente atribuídas a interrupções de uso geral.
  - **Solicitações de interrupção rápida** - são normalmente reservadas para uma única fonte de interrupção que requer um tempo de resposta rápido.

Software Interrupt Instruction

```
Syntax: SWI{<cond>} SWI_number
```

SWI	software interrupt	*lr_svc* = address of instruction following the SWI *spsr_svc* = *cpsr* *pc* = vectors + 0x8 *cpsr* mode = *SVC* *cpsr I* = 1 (mask IRQ interrupts)

```
SWI_Number = <SWI instruction> AND NOT(0xff000000)
```



Software Interrupt Instruction

```
SWI_handler
        ;
        ; Store registers r0-r12 and the link register
        ;
        STMFD   sp!, {r0-r12, lr}

        ; Read the SWI instruction
        LDR       r10, [lr, #-4]

        ; Mask off top 8 bits
        BIC       r10, r10, #0xff000000

        ; r10 - contains the SWI number
        BL        service_routine

        ; return from SWI handler
        LDMFD   sp!, {r0-r12, pc}^
```

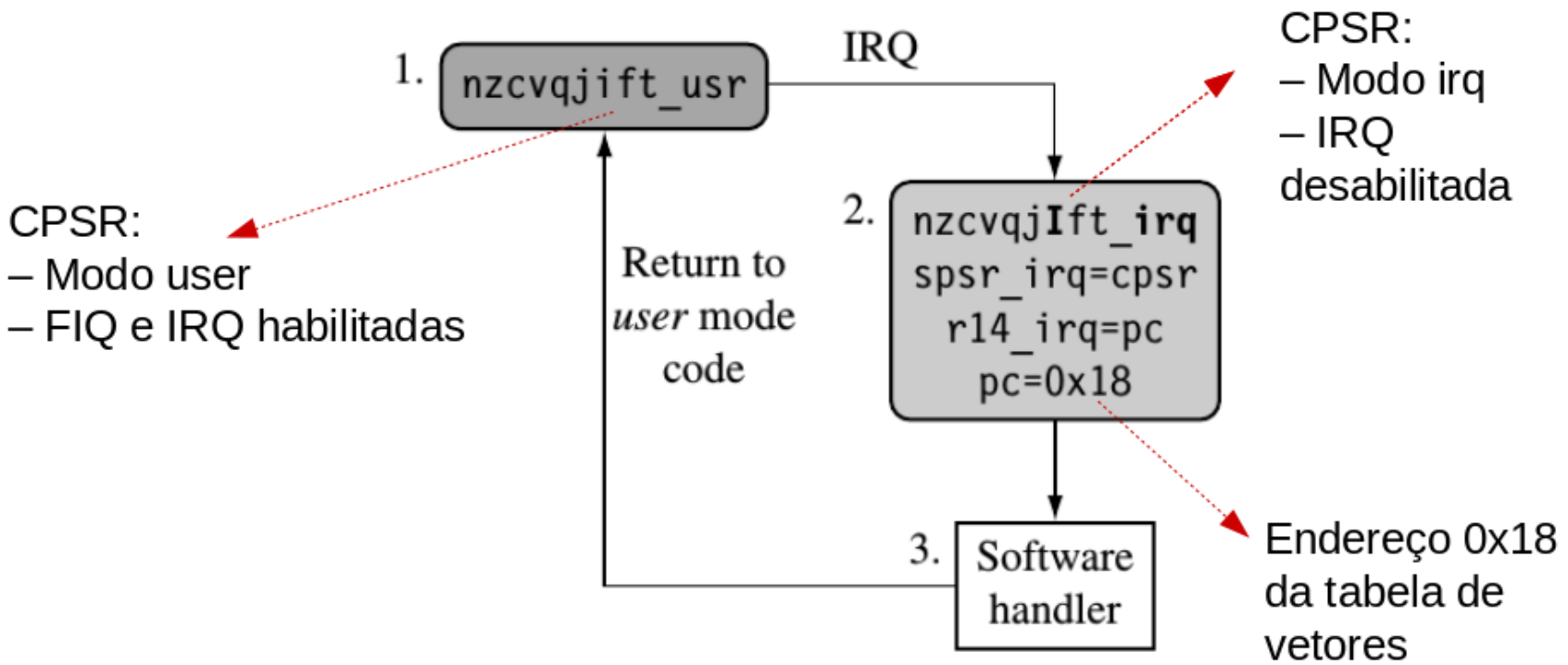
Exceções de IRQ e FIQ

- As exceções de IRQ e FIQ ocorrem apenas quando uma máscara de interrupção específica é limpa no $cpsr$.
- O processador ARM continuará executando a instrução atual no estágio de execução do pipeline antes de manipular a interrupção.
- O procedimento varia um pouco, dependendo do tipo de interrupção que está sendo disparada.

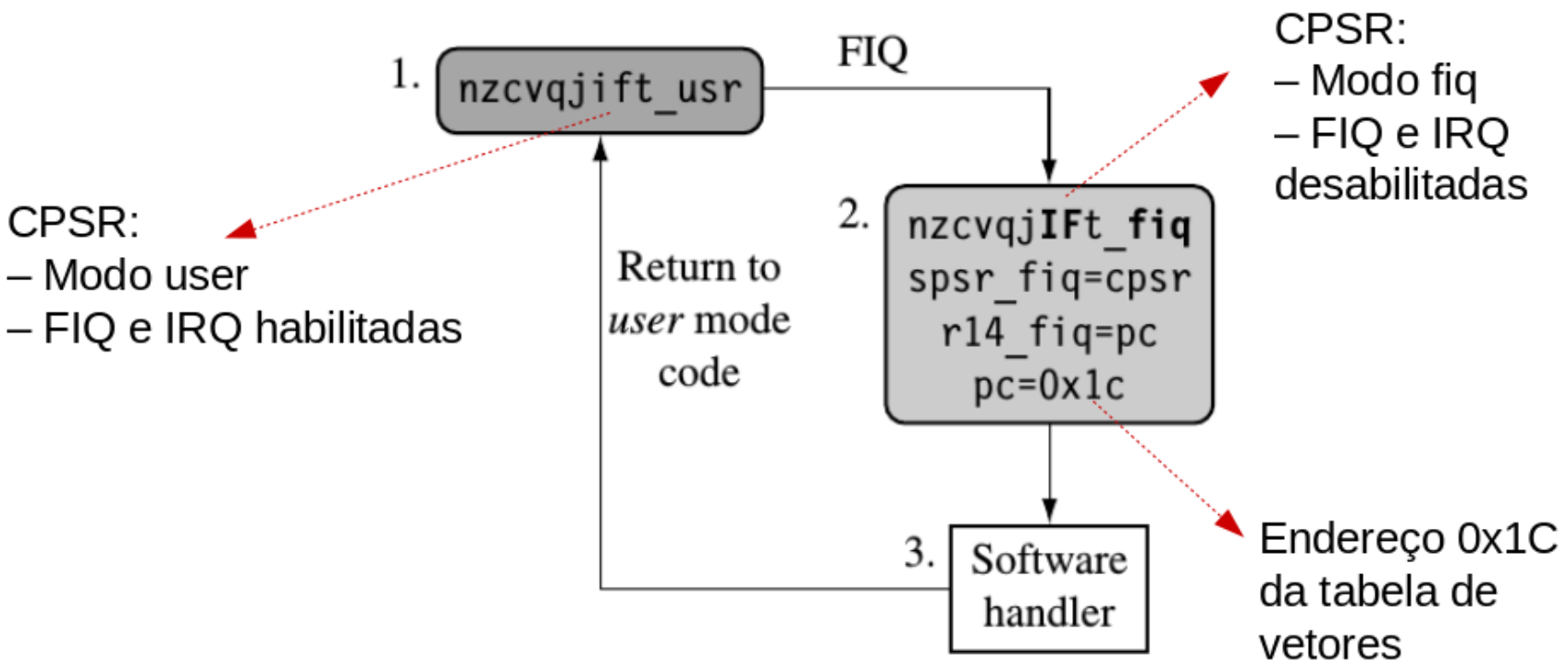
Exceções de IRQ e FIQ

- Uma exceção IRQ ou FIQ faz com que o hardware do processador passe por um procedimento padrão:
  - O processador muda para um modo de solicitação de interrupção específico, que reflete a interrupção que está sendo disparada.
  - O $cpsr$ do modo anterior é salvo no $spsr$ do novo modo de solicitação de interrupção.
  - O PC é salvo no lr do novo modo de solicitação de interrupção.
  - As interrupções estão desabilitadas - as exceções IRQ ou IRQ e FIQ estão desabilitadas no $cpsr$. Isso imediatamente interrompe a solicitação de outra interrupção do mesmo tipo.
  - O processador ramifica para uma entrada específica na tabela de vetores.

Interrupt Request



Fast Interrupt Request



Habilitando/desabilitando interrupções

Enabling an interrupt.

cpsr value	IRQ	FIQ
Pre	*nzcvqj**I**Ft_SVC*	*nzcvqj**I**Ft_SVC*

Code	`enable_irq`
	`MRS r1, cpsr`
	`BIC r1, r1, #0x80`
	`MSR cpsr_c, r1`
		`enable_fiq`
		`MRS r1, cpsr`
		`BIC r1, r1, #0x40`
		`MSR cpsr_c, r1`

Post	*nzcvqj**i**Ft_SVC*	*nzcvqj**I**ft_SVC*

Disabling an interrupt.

cpsr	IRQ	**FIQ**
Pre	*nzcvqj**i**ft_SVC*	*nzcvqj**i**ft_SVC*

Code	`disable_irq`
	`MRS r1, cpsr`
	`ORR r1, r1, #0x80`
	`MSR cpsr_c, r1`
		`disable_fiq`
		`MRS r1, cpsr`
		`ORR r1, r1, #0x40`
		`MSR cpsr_c, r1`

Post	*nzcvqj**I**ft_SVC*	*nzcvqj**i**Ft_SVC*

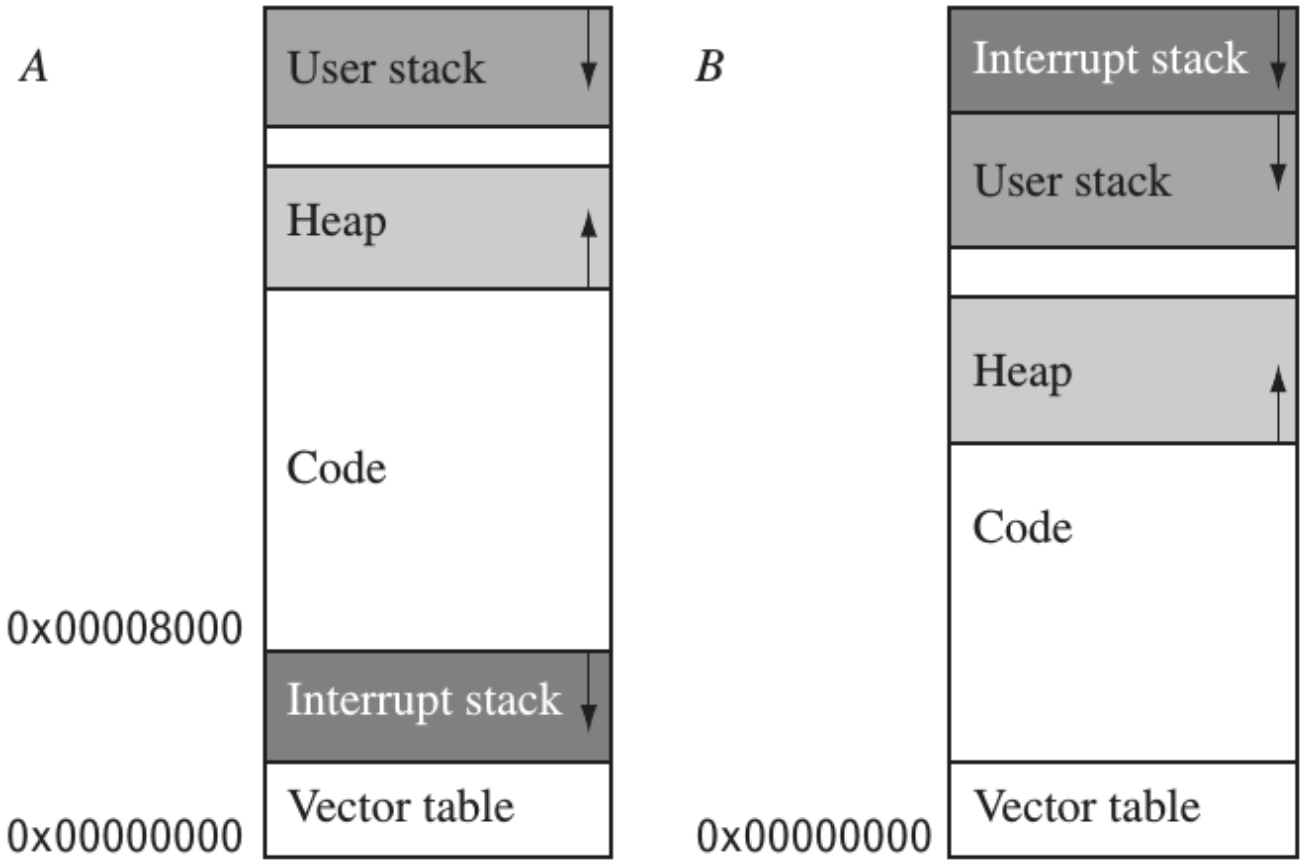
Projeto e Implementação Básica da Pilha de Interrupções

- Os manipuladores de exceções fazem uso extensivo de pilhas, com cada modo tendo um registro dedicado contendo o ponteiro da pilha.
- O design das pilhas de exceção depende desses fatores:
  - **Requisitos do sistema operacional** - Cada sistema operacional possui seus próprios requisitos para o design da pilha.
  - **Hardware de destino** - O hardware de destino fornece um limite físico para o tamanho e o posicionamento da pilha na memória.
- Decisões que precisam ser tomadas no projeto da pilha
  - O **local** determina onde no mapa de memória a pilha começa.
  - O **tamanho** da pilha depende do tipo de manipulador, aninhado ou não.

Projeto e Implementação Básica da Pilha de Interrupções

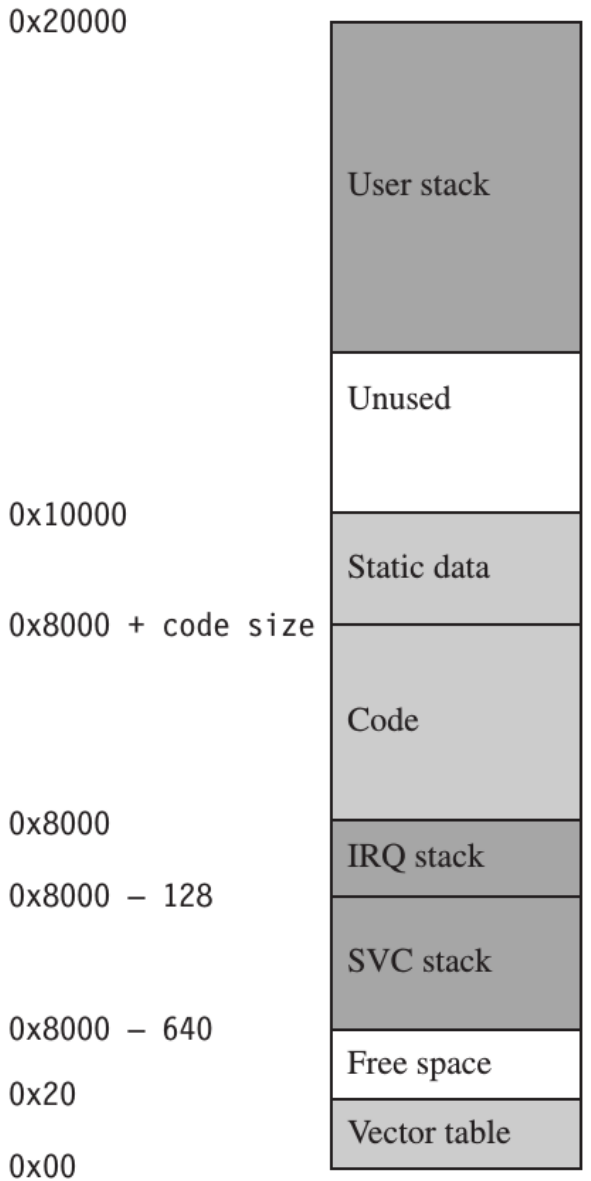
- Um bom projeto de pilha tenta evitar o estouro de pilha.
- Existem técnicas de software que identificam o estouro e permitem que medidas corretivas sejam tomadas para reparar a pilha antes que ocorra uma corrupção irreparável da memória.
  - usar proteção de memória;
  - chamar uma função de verificação de pilha no início de cada rotina.

Stack layouts



- Exemplo:

```
USR_Stack       EQU 0x20000
IRQ_Stack       EQU 0x8000
SVC_Stack       EQU IRQ_Stack-128
```



Example implementation using layout *A*.

Exemplo

```
USR_Stack       EQU 0x20000
IRQ_Stack       EQU 0x8000
SVC_Stack       EQU IRQ_Stack-128

Usr32md         EQU 0x10                ; User mode
FIQ32md         EQU 0x11                ; FIQ mode
IRQ32md         EQU 0x12                ; IRQ mode
SVC32md         EQU 0x13                ; Supervisor mode
Abt32md         EQU 0x17                ; Abort mode
Und32md         EQU 0x1b                ; Undefined instruction mode
Sys32md         EQU 0x1f                ; System mode

NoInt           EQU 0xc0                ; Disable interrupts
```

Exemplos

// Configurar pilha do modo SVC

```
        LDR     r13, SVC_NewStack          ; r13_svc
        ...
SVC_NewStack
        DCD   SVC_Stack
```

// Configurar pilha do modo IRQ

```
        MOV     r2, #NoInt|IRQ32md
        MSR     cpsr_c, r2
        LDR     r13, IRQ_NewStack          ; r13_irq
        ...
IRQ_NewStack
        DCD     IRQ_Stack
```

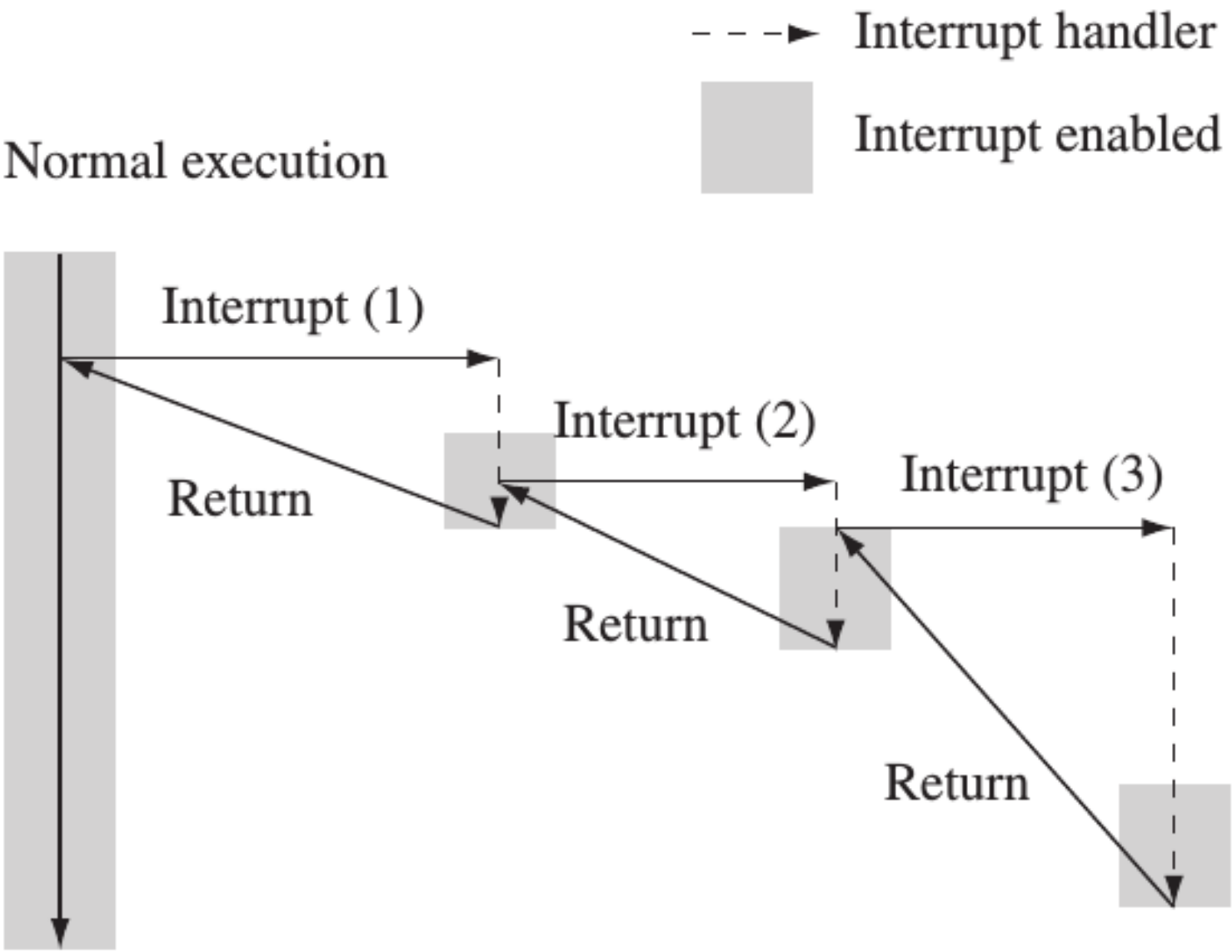
Latência das Interrupções

- A latência de interrupção depende de uma combinação de hardware e software.
- Os arquitetos do sistema devem equilibrar o design do sistema para manipular várias fontes de interrupção simultâneas e minimizar a latência de interrupção.
- Se as interrupções não forem tratadas em tempo hábil, o sistema exibirá tempos de resposta lentos.

Latência das Interrupções

- Os manipuladores de software têm dois métodos principais para minimizar a latência de interrupção.
  - O primeiro método é usar um manipulador de interrupções aninhado (*nested interrupt handler*), o que permite que outras interrupções ocorram enquanto está atendendo uma interrupção existente.
  - O segundo método envolve priorização. Você programa o controlador de interrupção para ignorar interrupções da mesma ou menor prioridade que a interrupção que você está manipulando.
- O processador gasta tempo nas interrupções de prioridade mais baixa até que ocorra uma interrupção de prioridade mais alta.
- Portanto, as interrupções de prioridade mais alta têm uma latência de interrupção média mais baixa do que as interrupções de prioridade mais baixa.

Latência das interrupções



QXD0133 - Arquitetura e Organização de Computadores II

Universidade Federal do Ceará - Campus Quixadá

Thiago Werlley
thiagowerlley@ufc.br

18 de outubro de 2025

Capítulo 9