Interrupções e Exceções

Universidade Federal do Ceará - Campus Quixadá

Roberto Cabral
rbcabral@ufc.br

03 de Março de 2021

Arquitetura e Organização de Computadores II

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

- Uma exceção é qualquer condição que precise interromper a execução sequencial normal de instruções.
- O tratamento de exceções é o método de processamento dessas exceções.
- A maioria das exceções possui um tratamento de exceção de software associado
  - uma rotina de software que é executada quando ocorre uma exceção.

Exceções e modos do processador ARM

- Cada exceção faz com que o processador entre em um modo específico.
- Além disso, qualquer um dos modos do processador ARM pode ser inserido manualmente, alterando o $cpsr$.
- O usuário e o modo do sistema são os únicos dois modos que não são inseridos por uma exceção correspondente, ou seja, para entrar nesses modos, você deve modificar o $cpsr$.
- Quando uma exceção causa uma mudança de modo, o processador automaticamente
  - salva o $cpsr$ no $spsr$ do modo de exceção
  - salva o PC no lr do modo de exceção
  - define o $cpsr$ para o modo de exceção
  - define pc no endereço do manipulador de exceções

Exceções e modos associados

Exception	Mode	Main purpose
Fast Interrupt Request	*FIQ*	fast interrupt request handling
Interrupt Request	*IRQ*	interrupt request handling
SWI and Reset	*SVC*	protected mode for operating systems
Prefetch Abort and Data Abort	*abort*	virtual memory and/or memory protection
Undefined Instruction	*undefined*	software emulation of hardware coprocesso

Exceções e modos associados

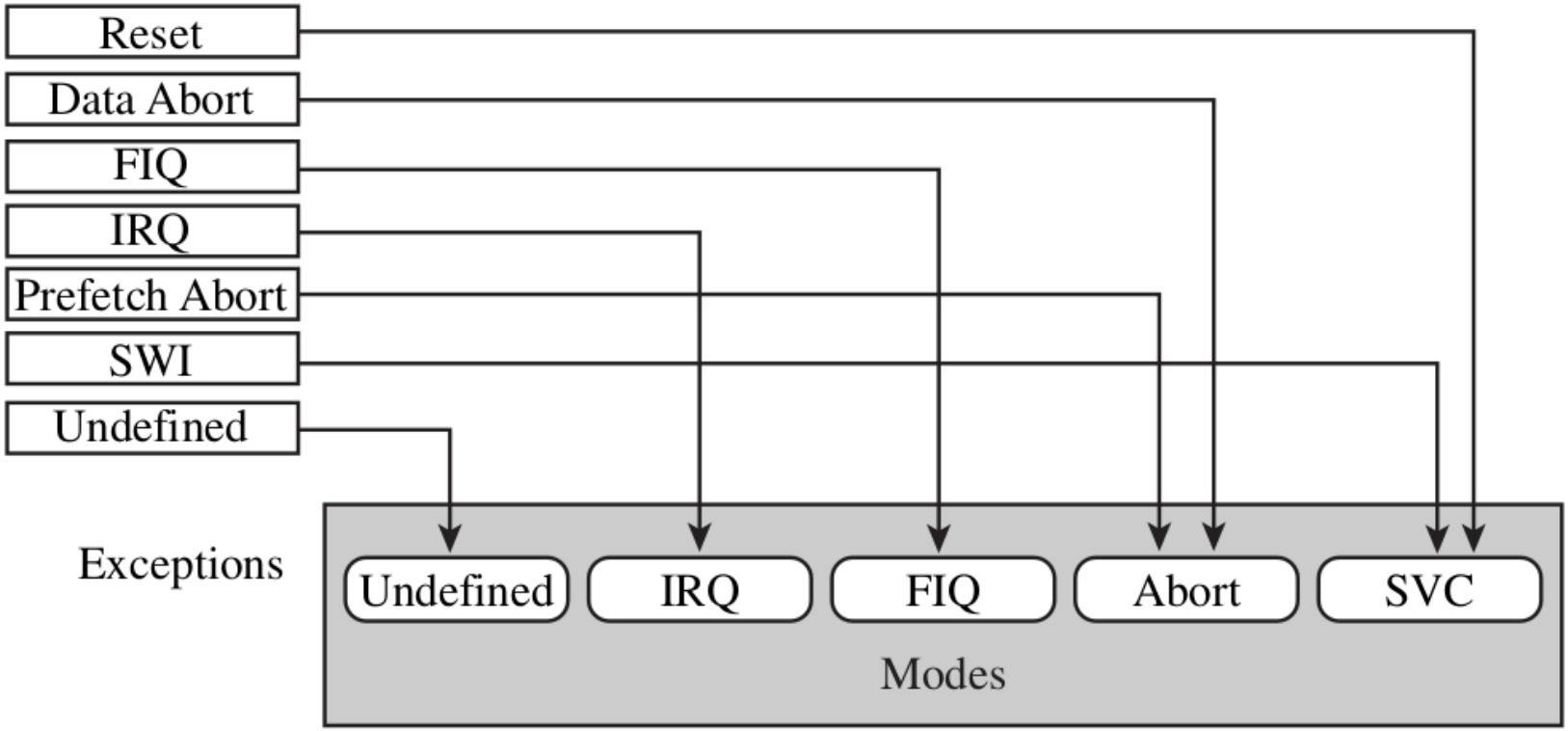
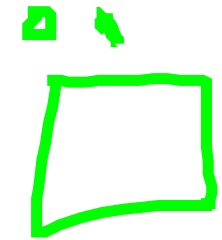


Tabela de Vetores

- 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 0xfffff030 para o PC.
  - `MOV pc, #imediato` - copia um valor imediato para o PC.

Tabela de vetores e modos do processador

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



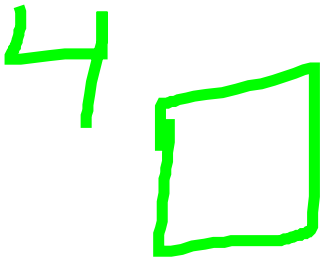
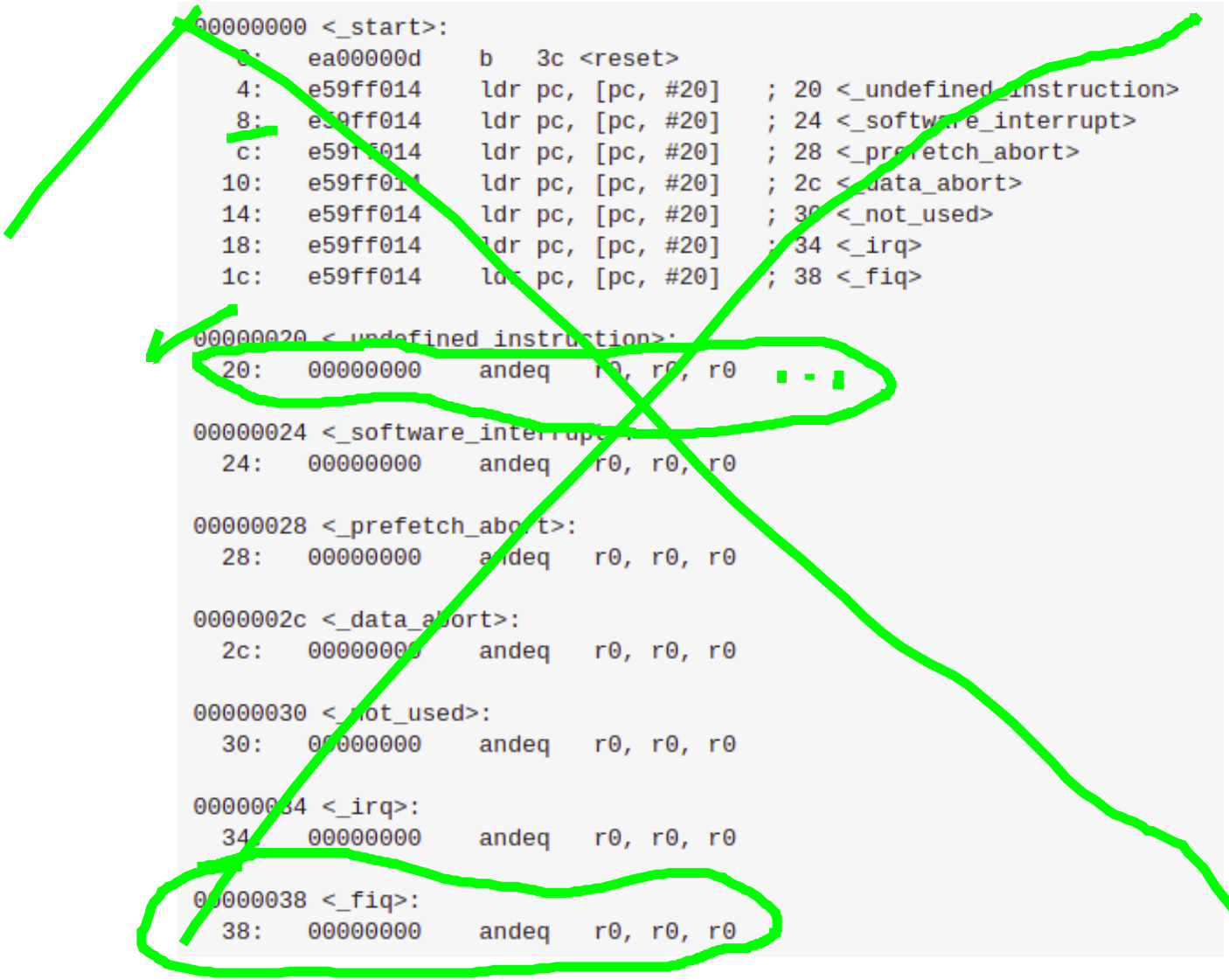


Tabela de vetores - Exemplo



```
00000000 <_start>:
   0:   ea00000d    b   3c <reset>
   4:   e59ff014    ldr pc, [pc, #20]   ; 20 <_undefined_instruction>
   8:   e59ff014    ldr pc, [pc, #20]   ; 24 <_software_interrupt>
   c:   e59ff014    ldr pc, [pc, #20]   ; 28 <_prefetch_abort>
  10:   e59ff014    ldr pc, [pc, #20]   ; 2c <_data_abort>
  14:   e59ff014    ldr pc, [pc, #20]   ; 30 <_not_used>
  18:   e59ff014    ldr pc, [pc, #20]   ; 34 <_irq>
  1c:   e59ff014    ldr pc, [pc, #20]   ; 38 <_fiq>

00000020 <_undefined_instruction>:
  20:   00000000    andeq   r0, r0, r0

00000024 <_software_interrupt>:
  24:   00000000    andeq   r0, r0, r0

00000028 <_prefetch_abort>:
  28:   00000000    andeq   r0, r0, r0

0000002c <_data_abort>:
  2c:   00000000    andeq   r0, r0, r0

00000030 <_not_used>:
  30:   00000000    andeq   r0, r0, r0

00000034 <_irq>:
  34:   00000000    andeq   r0, r0, r0

00000038 <_fiq>:
  38:   00000000    andeq   r0, r0, r0
```

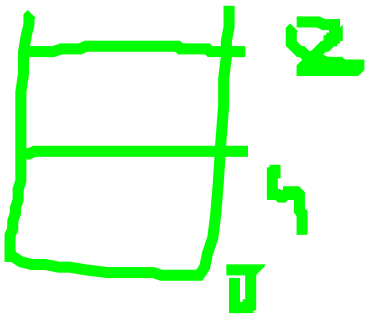
Prioridades de Exceção

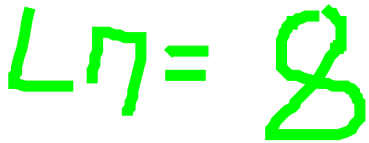
- Exceções podem ocorrer simultaneamente, portanto, o processador precisa adotar um mecanismo de prioridade.

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	—

Offsets de registro de link

- 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 $lr - 4$.

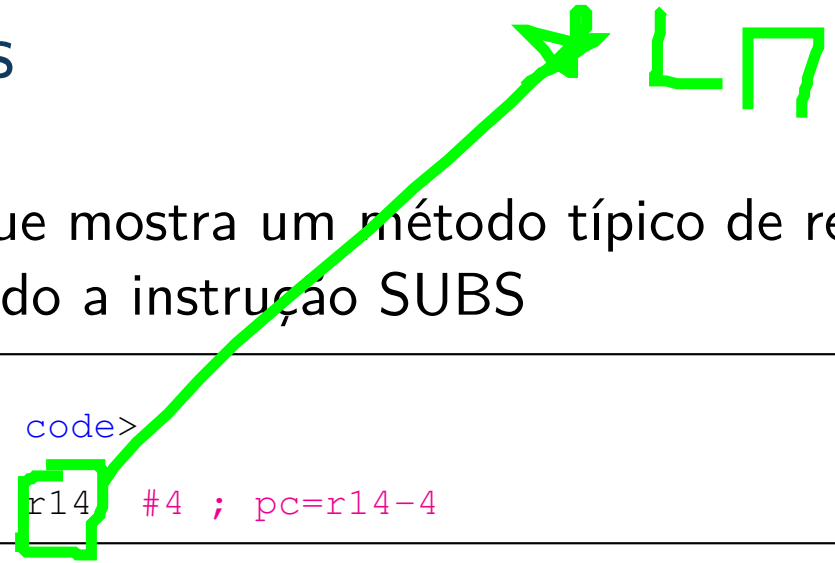




Useful link-register-based addresses.

Exception	Address	Use
Reset	—	*lr* is not defined on a Reset
Data Abort	$lr - 8$	points to the instruction that caused the Data Abor
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 A
SWI	*lr*	points to the next instruction after the SWI instruc
Undefined Instruction	*lr*	points to the next instruction after the undefined ir

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:
  - As interrupções de software - são normalmente reservadas para chamar rotinas privilegiadas do sistema operacional.
  - Solicitações de interrupção - são normalmente atribuídas a interrupções de uso geral.
  - As solicitações de interrupção rápida - são normalmente reservadas para uma única fonte de interrupção que requer um tempo de resposta rápido.

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.

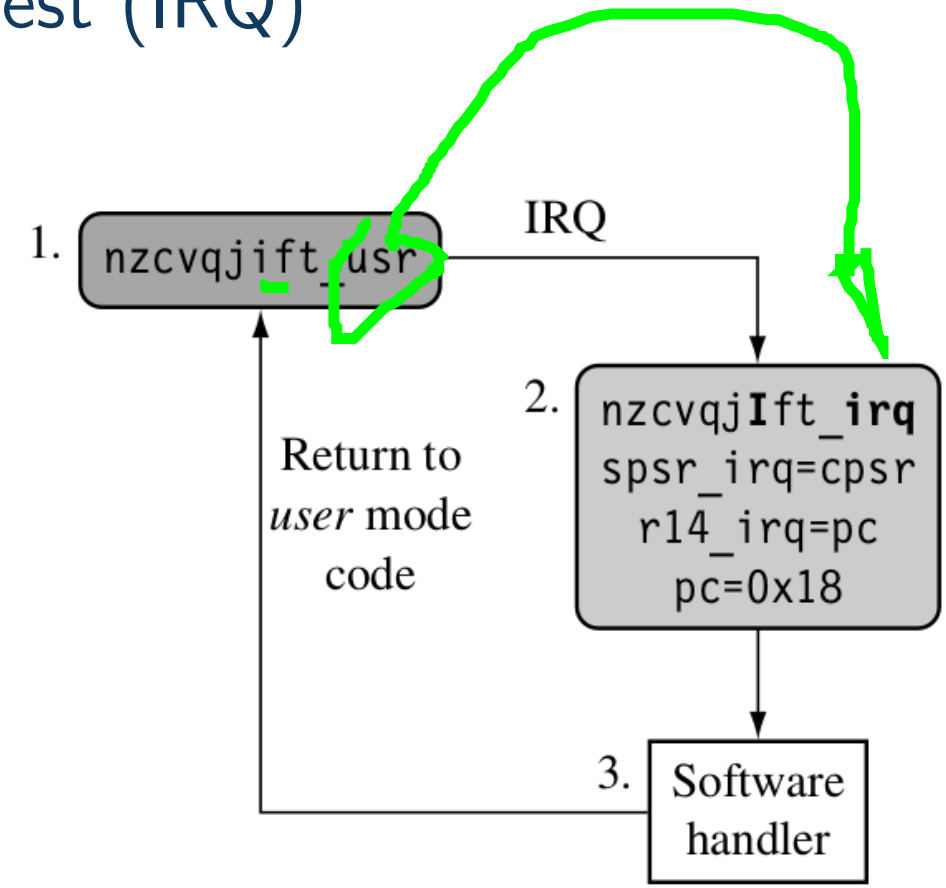
Exceções de IRQ e FIQ

- 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.

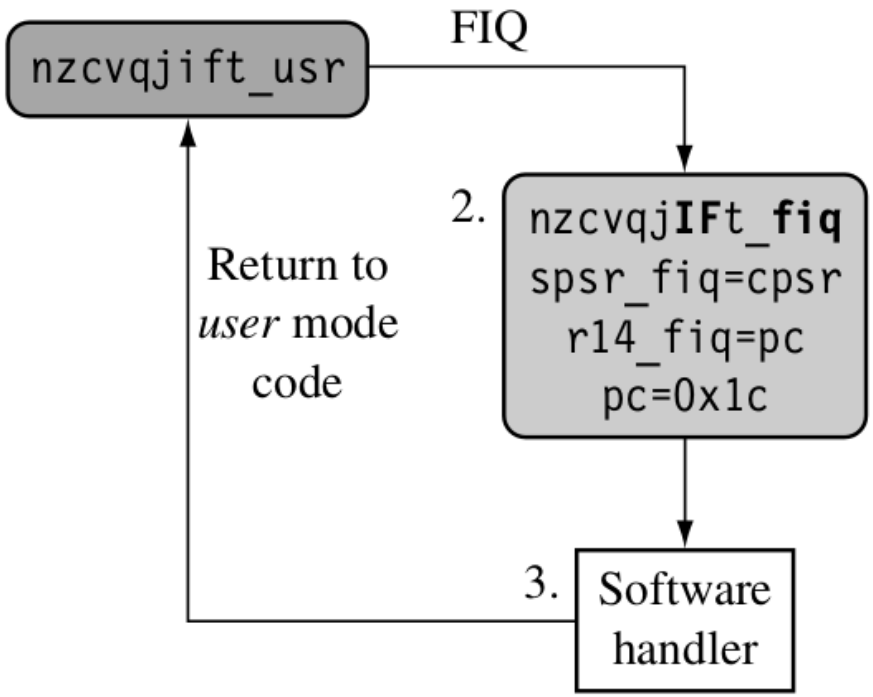
Exceções de IRQ e FIQ

- 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 (IRQ)



Fast Interrupt Request (FIQ)



Interrupt Request (IRQ)

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*	*nzcvqji**F**t_SVC*

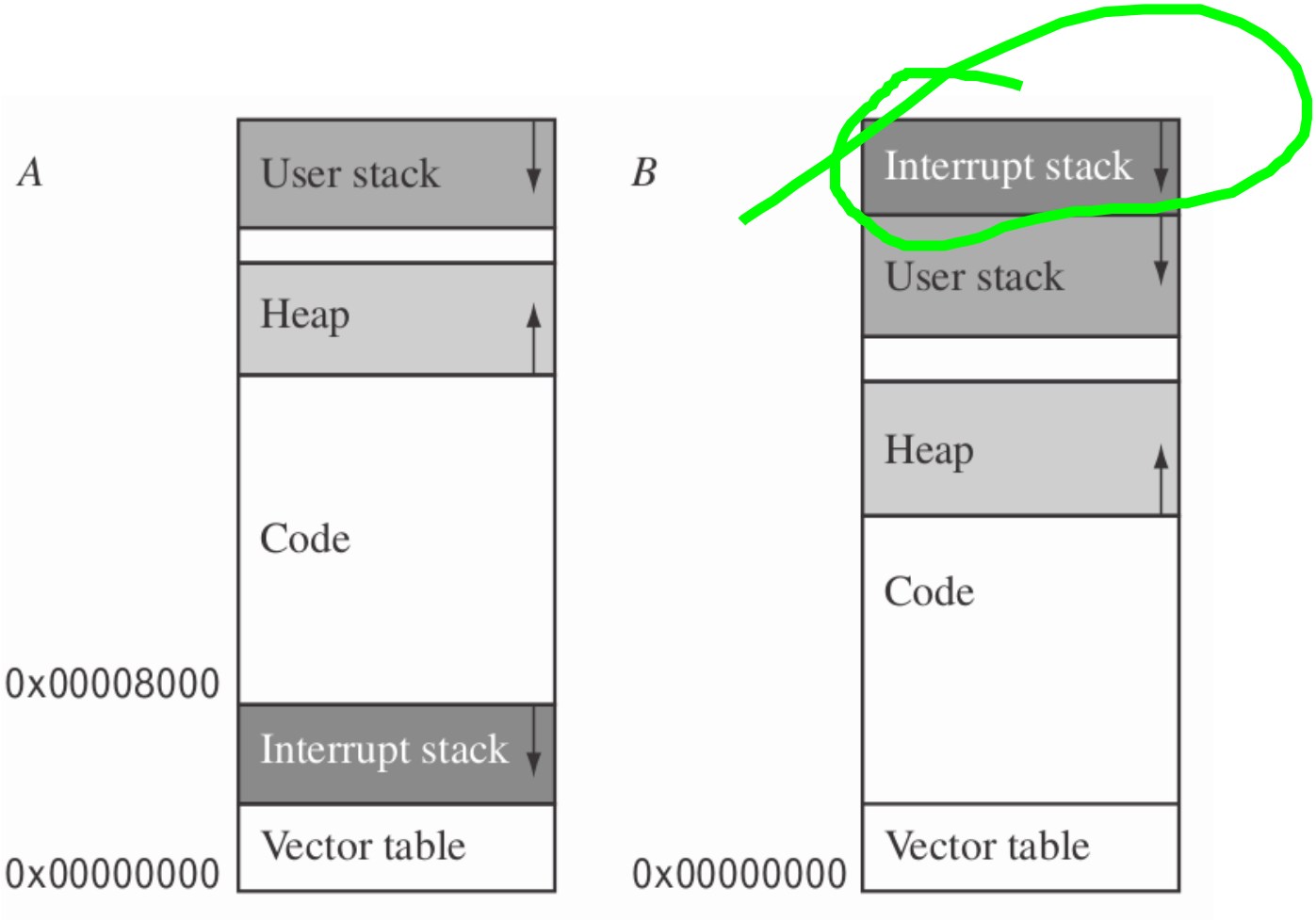
Projeto e Implementação Básica da Pilha de Interrupções

- 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.

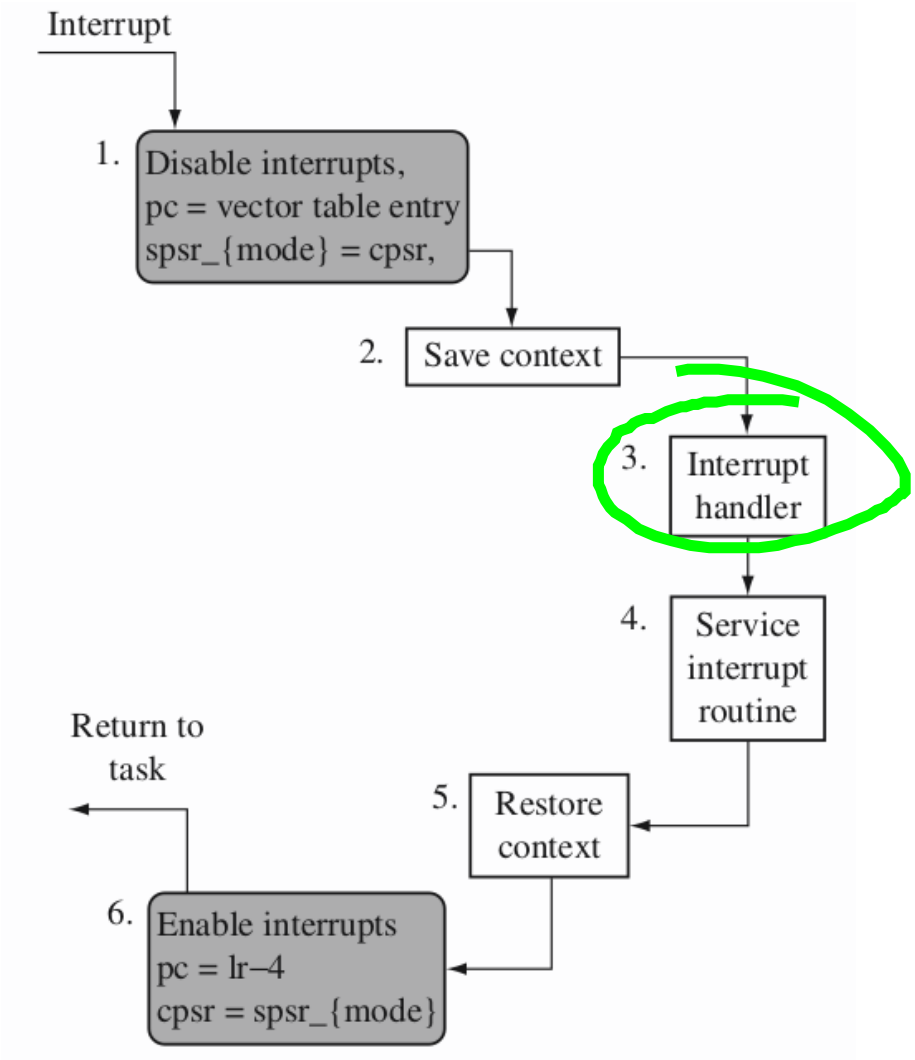
Typical memory layouts



Manipulador de interrupção não aninhado

- É o manipulador de interrupção mais simples.
- As interrupções são desativadas até o controle retornar à tarefa ou processo interrompido.
- Como um manipulador de interrupção não aninhado só pode atender uma única interrupção por vez, os manipuladores deste formulário não são adequados para sistemas embarcados complexos que atendem a várias interrupções com diferentes níveis de prioridade.

Manipulador de interrupção não aninhado



Manipulador de interrupção não aninhado

- *Disable Interrupt*:
  - desabilita a ocorrência de outras exceções
  - faz o backup do cpsr
  - define o modo de solicitação de interrupção
  - ajusta o PC para apontar para a entrada correta na tabela de vetores
- *Save context* - salva um subconjunto dos registradores.
- *Interrupt handler* - identifica a fonte de interrupção externa e executa a rotina de serviço de interrupção (ISR) apropriada.
- *Interrupt service routine* - o ISR atende a fonte de interrupção externa e redefine a interrupção.
- *Restore context* - O ISR retorna ao manipulador de interrupções, que restaura o contexto.
- *Enable interruption* - restaura o cpsr e ajusta o PC para a próxima instrução após a interrupção ter sido disparada.

Manipulador de interrupção não aninhado

```
interrupt_handler
    SUB r14,r14,#4 ; adjust lr
    STMFD r13!,{r0-r3,r12,r14} ; save context
    <interrupt service routine>
    LDMFD r13!,{r0-r3,r12,pc}* ; return
```

```
interrupt_handler
    SUB r14,r14,#4 ; r14-=4

    STMFD sp!,{r0-r3,r12,r14}; save context

    LDR r0,=IRQStatus; interrupt status addr

    LDR r0,[r0]; get interrupt status

    TST r0,#0x0080; if counter timer

    BNE timer_isr; then branch to ISR

    TST r0,#0x0001; else if button press

    BNE button_isr; then call button ISR

    LDMFD sp!,{r0-r3,r12,r14}; restore context

    LDR pc,=debug_monitor; else debug monitor
```

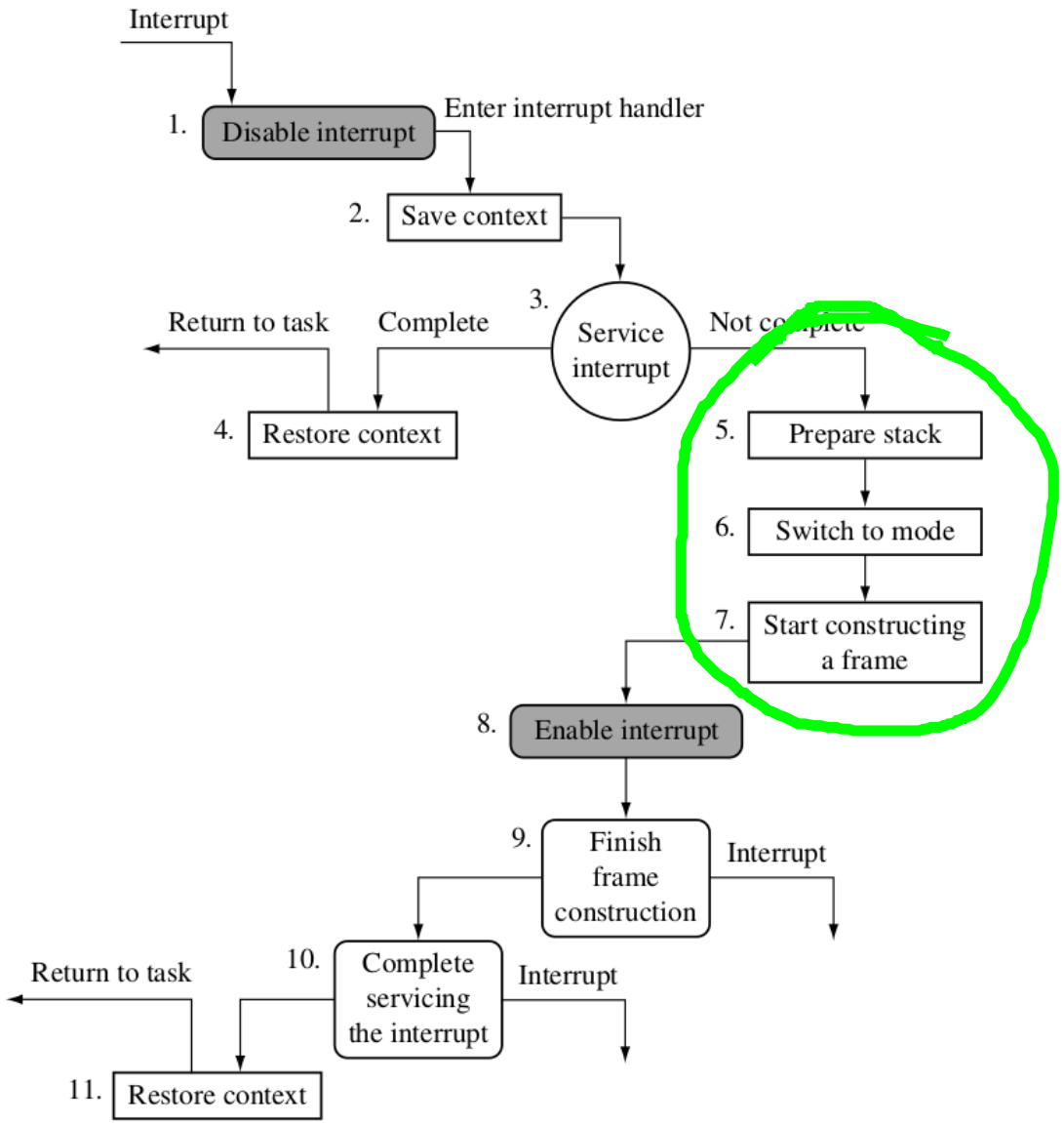
Manipulador de interrupção não aninhado

- Única interrupção sequencial
- Alta latência de interrupção; não pode lidar com outras interrupções que ocorrem enquanto uma interrupção está sendo tratada.
- Vantagens: relativamente fácil de implementar e depurar.
- Desvantagem: não pode ser usado para lidar com sistemas embarcados complexos com múltiplas interrupções de prioridade.

Manipulador de interrupção aninhado

- Um manipulador de interrupção aninhado permite que outra interrupção ocorra no manipulador atualmente chamado.
- Isso é alcançado reativando as interrupções antes que o manipulador atenda totalmente a interrupção atual.

Manipulador de interrupção aninhado



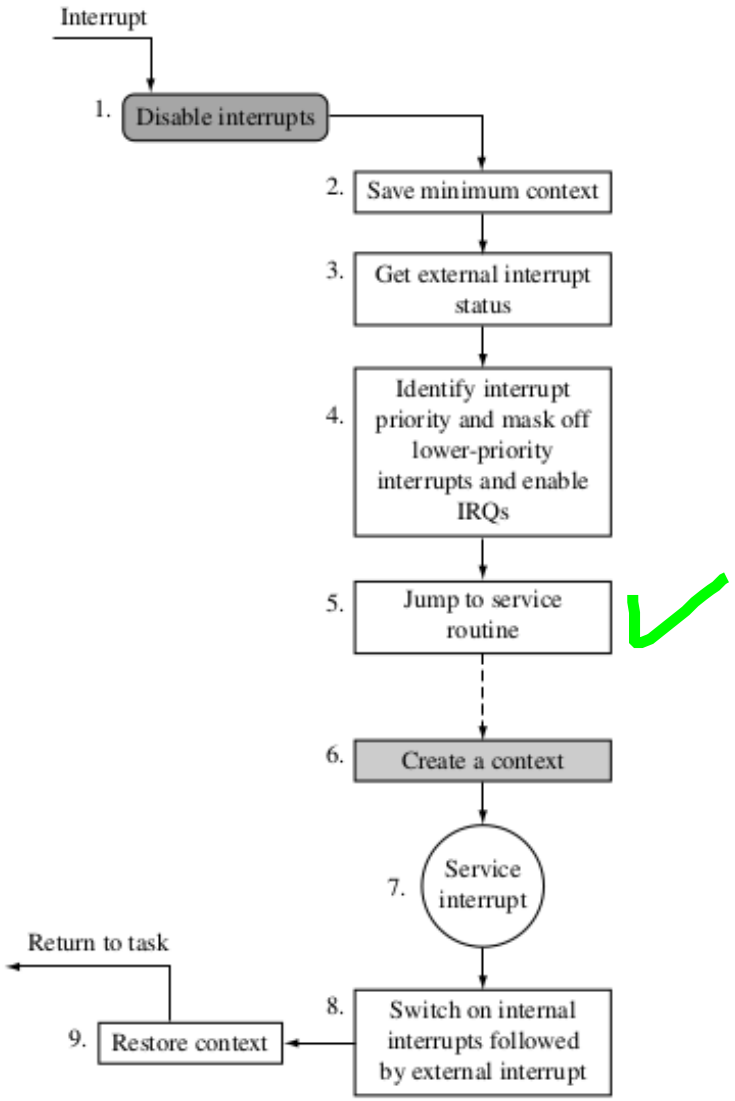
Manipulador de interrupção aninhado

- Lida com várias interrupções sem uma atribuição de prioridade.
- Latência de interrupção média a alta.
- Vantagem: pode ativar interrupções antes de terminar outras, reduzindo a latência da interrupção.
- Desvantagem: Não trata prioridades, assim uma interrupção de baixa prioridade pode bloquear uma de maior prioridade.

Manipulador de interrupção simples priorizado

- Lida com interrupções de prioridade mais alta em um tempo menor que as interrupções de prioridade mais baixa.
- Baixa latência de interrupção.
- Vantagem: interrupções de alta prioridade tratadas com maior urgência, sem duplicação de código, para definir máscaras de interrupção externas.
- Desvantagem: existe uma penalidade de tempo, pois esse manipulador exige dois saltos, resultando na liberação do pipeline cada vez que ocorre um salto.

Manipulador de interrupção simples priorizado



Outros esquemas

- Existem alguns outros esquemas, que são, na verdade, modificações aos esquemas anteriores, como segue:
  - Manipulador de interrupção reentrante - reativa as interrupções mais cedo e dá suporte às prioridades, para que a latência seja reduzida.
  - Manipulador de interrupção padrão priorizado - organiza as prioridades de uma maneira especial para reduzir o tempo necessário para decidir qual interrupção será tratada.
  - Manipulador de interrupções agrupadas priorizadas - agrupa algumas interrupções em um subconjunto que tem um nível de prioridade, isso é bom para uma grande quantidade de fontes de interrupção.

Resumo

- A disponibilidade de diferentes modos de operação no ARM ajuda no tratamento de exceções de forma estruturada.
- A troca de contexto é um dos principais problemas que afetam a latência de interrupção, e isso é resolvido no modo FIQ com o aumento do número de registradores armazenados.
- Não podemos decidir sobre um esquema de tratamento de interrupção para ser usado como um padrão em todos os sistemas, isso depende da natureza do sistema:
  - Que tipo de interrupções existem?
  - Quantas interrupções existem?

Interrupções e Exceções

Universidade Federal do Ceará - Campus Quixadá

Roberto Cabral
rbcabral@ufc.br

03 de Março de 2021

Arquitetura e Organização de Computadores II