QXD0133 - Arquitetura e Organização de Computadores II

Universidade Federal do Ceará - Campus Quixadá

Thiago Werlley
thiagowerlley@ufc.br

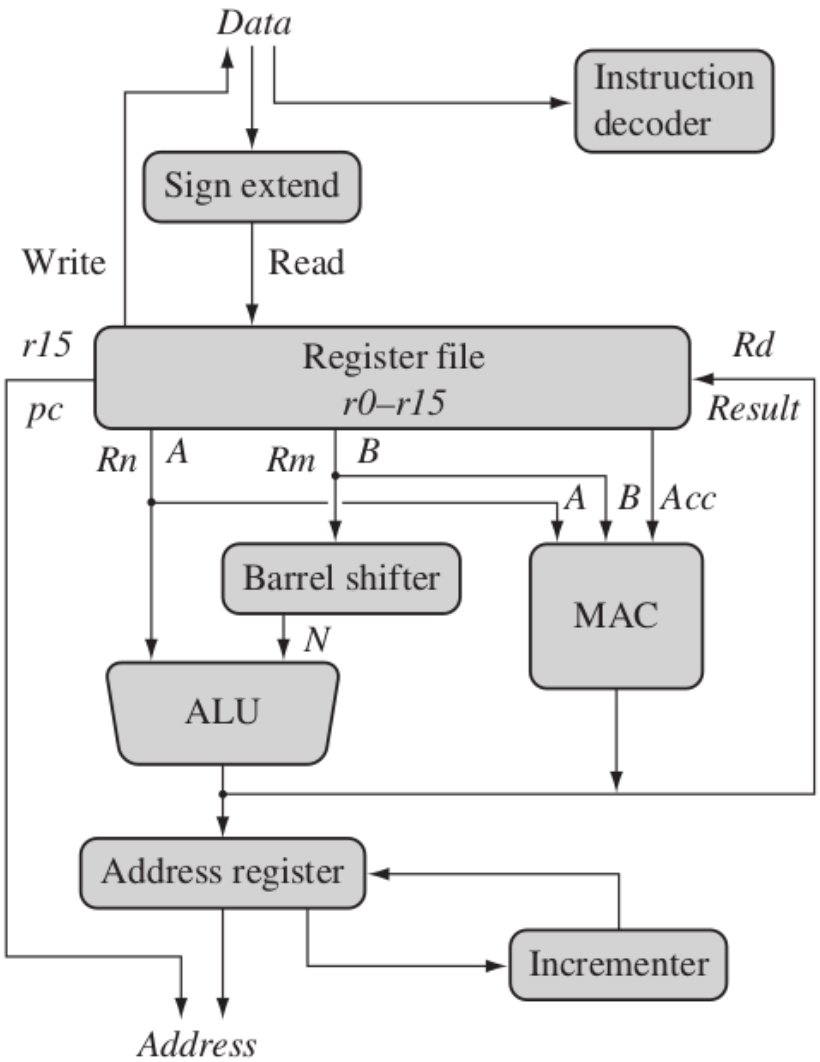
18 de outubro de 2025

Capítulo 2

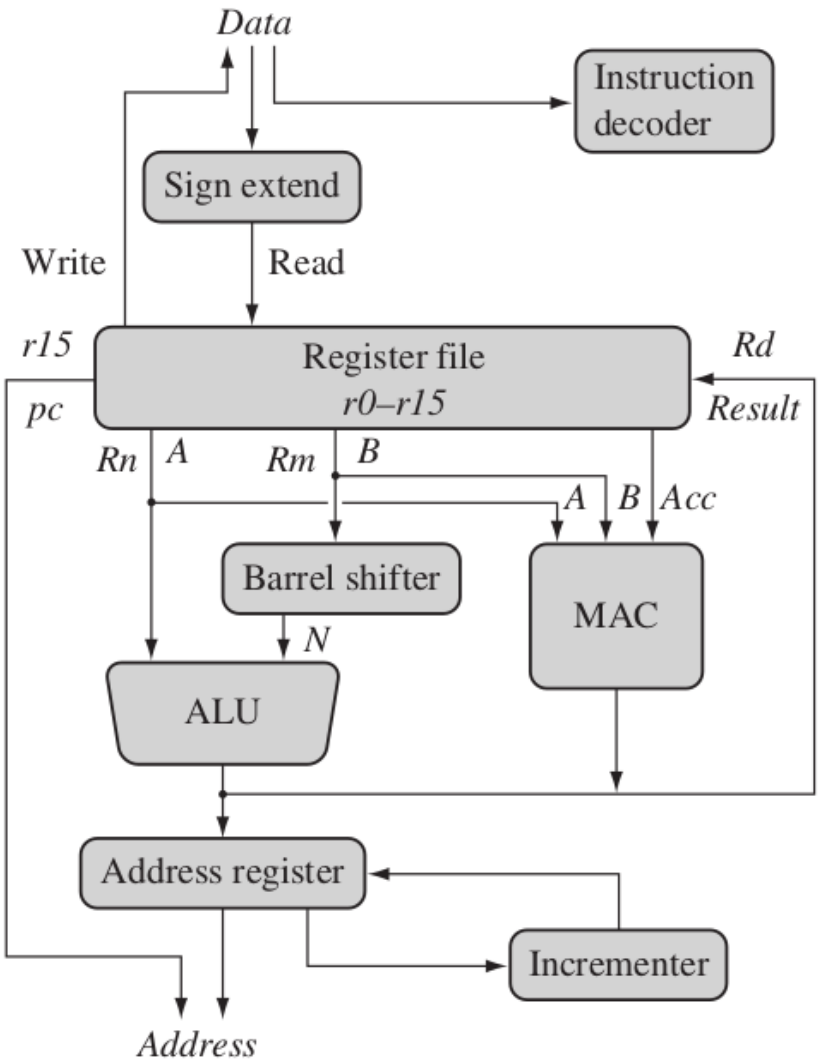
Capítulo 2

Fundamentos do processador ARM

ARM Core Dataflow Model

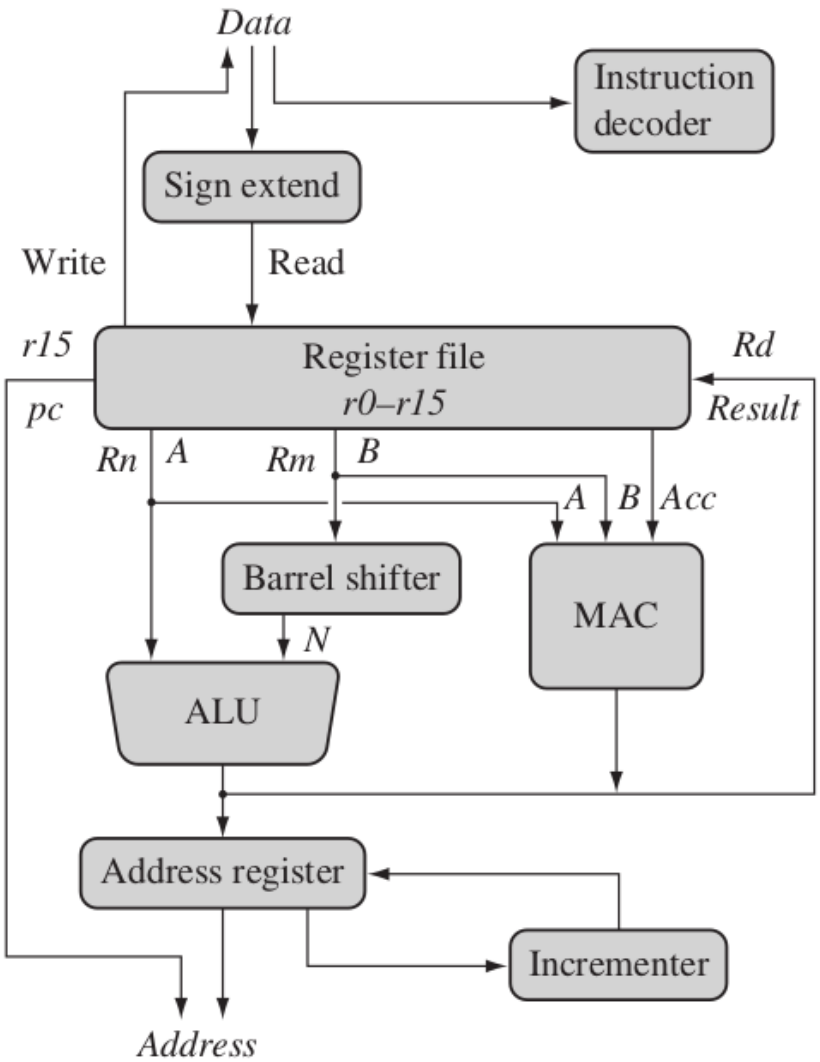


ARM Core Dataflow Model



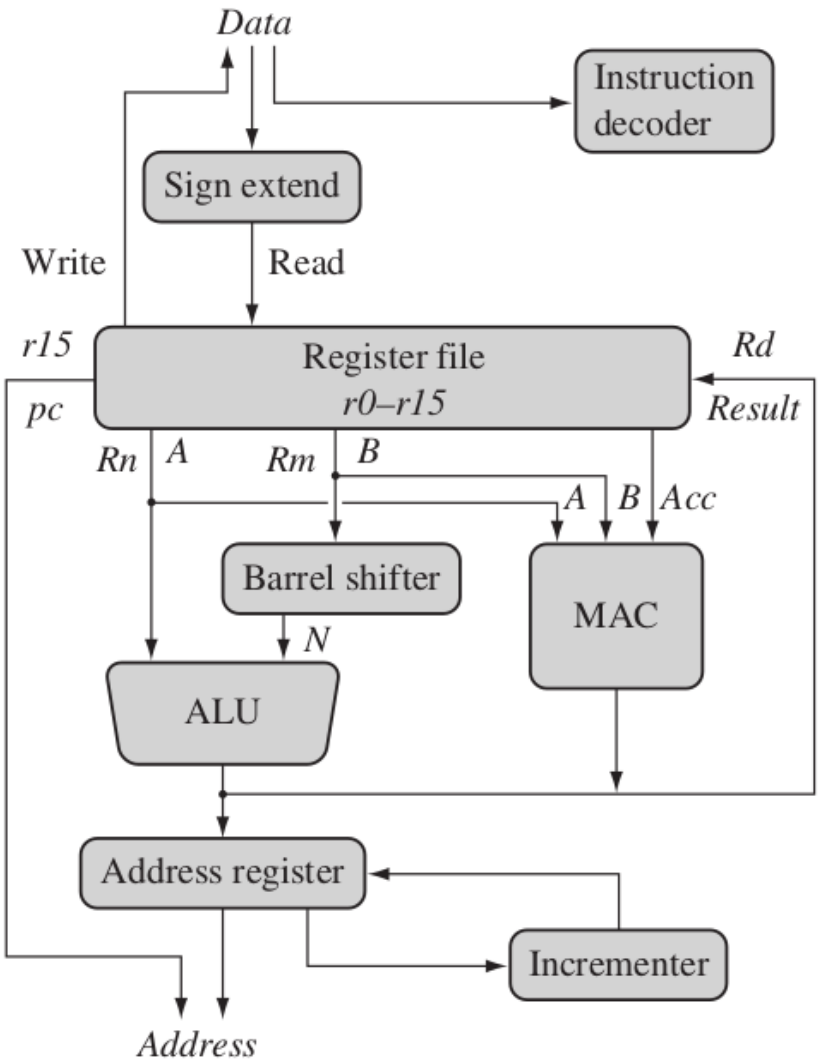
- Aquitetura **load-store**

ARM Core Dataflow Model



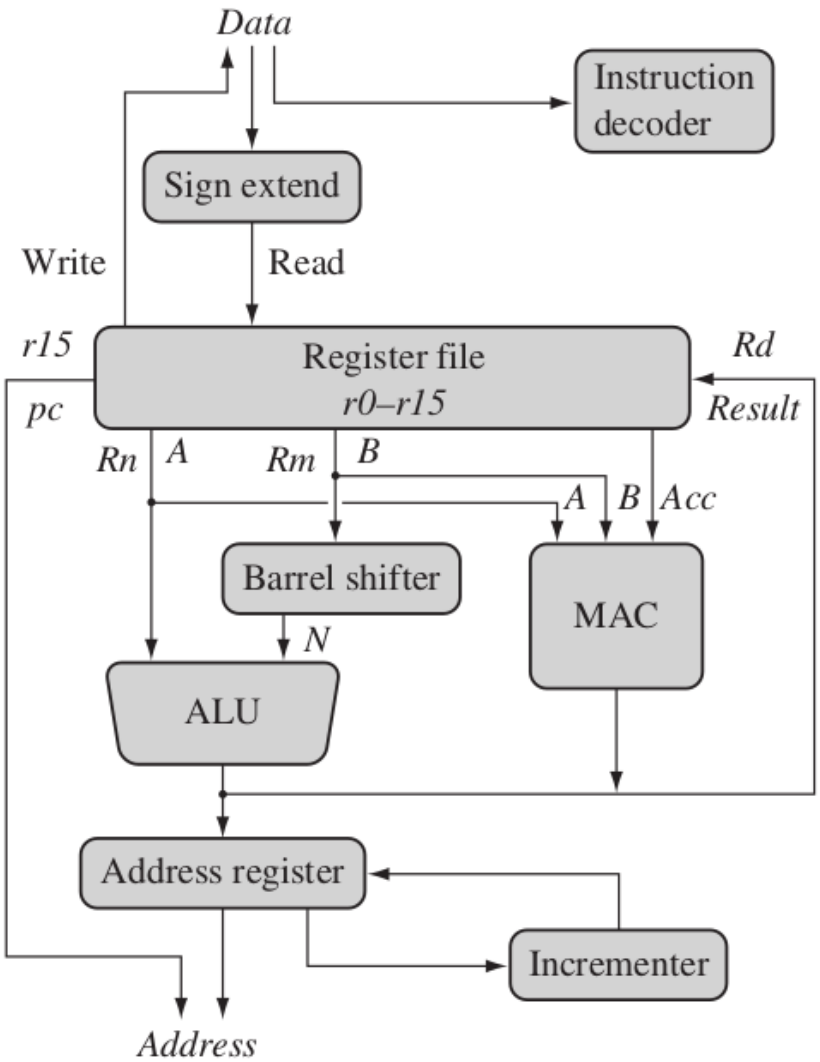
- Aquitetura **load-store**
  - **Load**: Memória para registrador

ARM Core Dataflow Model



- Aquitetura **load-store**
  - **Load**: Memória para registrador
  - **Store**: Registrador para memória

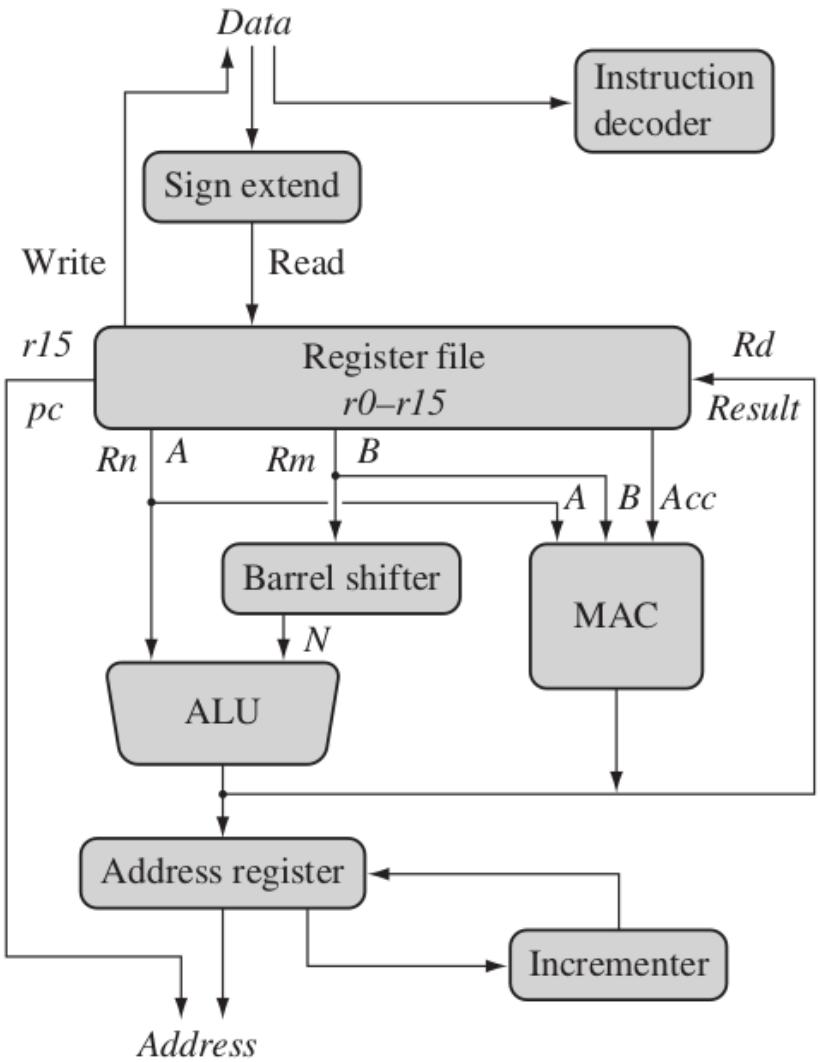
ARM Core Dataflow Model



- Aquitetura **load-store**
  - **Load**: Memória para registrador
  - **Store**: Registrador para memória

Sign extend: Conversão de dados sinalizados de 8 e 16 bits para dados sinalizados de 32 bits

ARM Core Dataflow Model

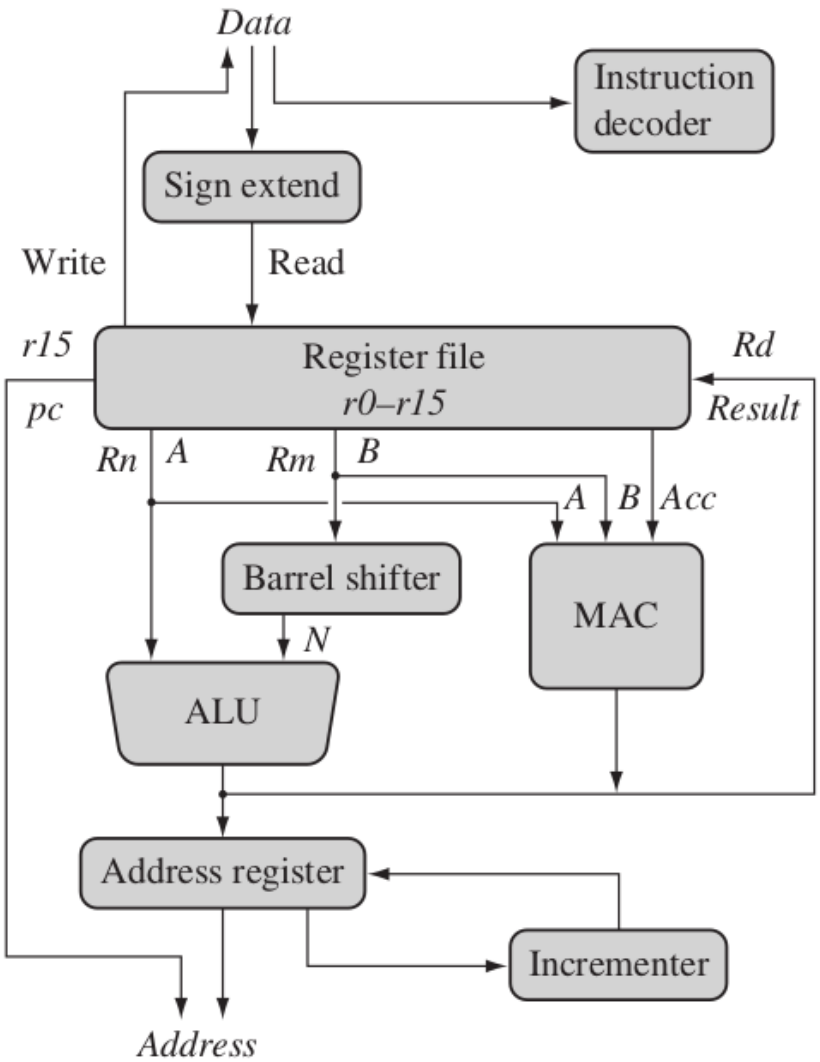


- Aquitetura **load-store**
  - **Load**: Memória para registrador
  - **Store**: Registrador para memória

Sign extend: Conversão de dados sinalizados de 8 e 16 bits para dados sinalizados de 32 bits

Rn, Rm $\rightarrow$ Operandos

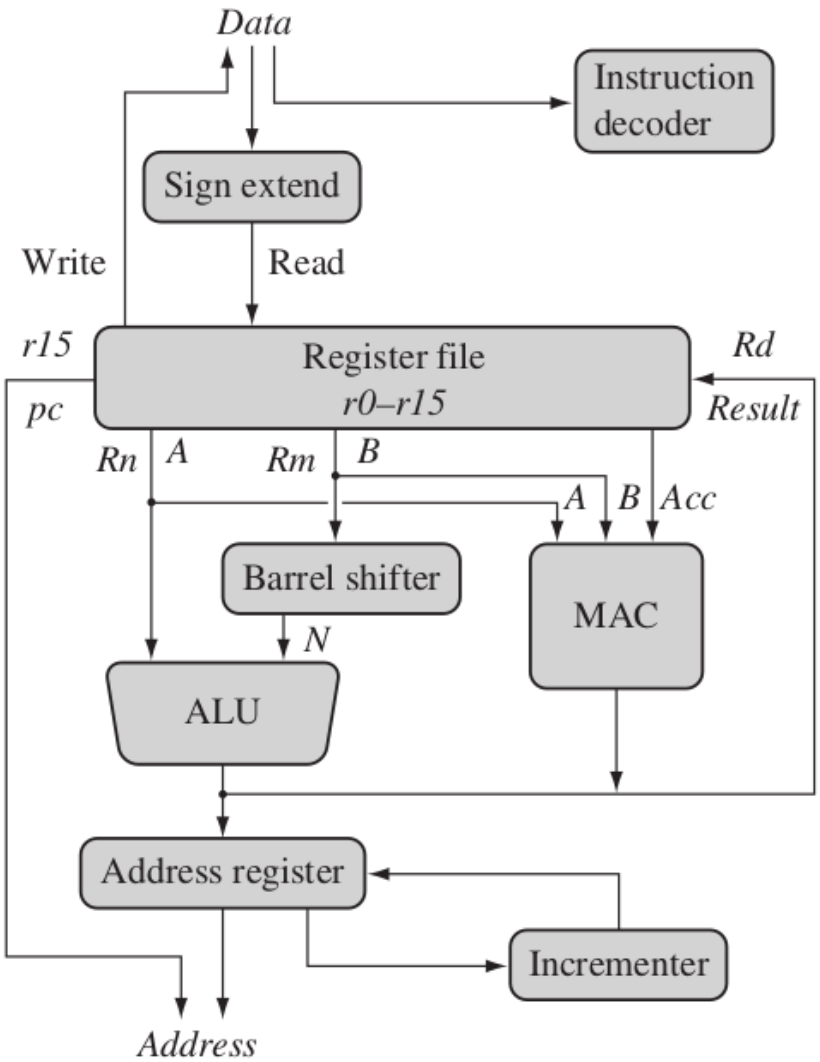
ARM Core Dataflow Model



- Aquitetura **load-store**
  - **Load**: Memória para registrador
  - **Store**: Registrador para memória

Sign extend: Conversão de dados sinalizados de 8 e 16 bits para dados sinalizados de 32 bits
Rn, Rm $\rightarrow$ Operandos
Rd $\rightarrow$ Resultado

ARM Core Dataflow Model



- Aquitetura **load-store**
  - **Load**: Memória para registrador
  - **Store**: Registrador para memória

Sign extend: Conversão de dados sinalizados de 8 e 16 bits para dados sinalizados de 32 bits
Rn, Rm → Operandos
Rd → Resultado
Barrel shifter →Deslocamento de dados em algumas instruções

Modos de operação (Processor Modes)

- User

Modos de operação (Processor Modes)

- User
  - Modo de execução normal

Modos de operação (Processor Modes)

- User
  - Modo de execução normal
- FIQ

Modos de operação (Processor Modes)

- User
  - Modo de execução normal
- FIQ
  - Interrupções rápidas, com alta velocidade

Modos de operação (Processor Modes)

- User
  - Modo de execução normal
- FIQ
  - Interrupções rápidas, com alta velocidade
- IRQ

Modos de operação (Processor Modes)

- User
  - Modo de execução normal
- FIQ
  - Interrupções rápidas, com alta velocidade
- IRQ
  - Interrupções gerais

Modos de operação (Processor Modes)

- User
  - Modo de execução normal
- FIQ
  - Interrupções rápidas, com alta velocidade
- IRQ
  - Interrupções gerais
- Supervisor

Modos de operação (Processor Modes)

- User
  - Modo de execução normal
- FIQ
  - Interrupções rápidas, com alta velocidade
- IRQ
  - Interrupções gerais
- Supervisor
  - Modo para o Sistema Operacional

Modos de operação (Processor Modes)

- User
  - Modo de execução normal
- FIQ
  - Interrupções rápidas, com alta velocidade
- IRQ
  - Interrupções gerais
- Supervisor
  - Modo para o Sistema Operacional
- Abort

Modos de operação (Processor Modes)

- User
  - Modo de execução normal
- FIQ
  - Interrupções rápidas, com alta velocidade
- IRQ
  - Interrupções gerais
- Supervisor
  - Modo para o Sistema Operacional
- Abort
  - Modo de proteção para o acesso à memória

Modos de operação (Processor Modes)

- User
  - Modo de execução normal
- FIQ
  - Interrupções rápidas, com alta velocidade
- IRQ
  - Interrupções gerais
- Supervisor
  - Modo para o Sistema Operacional
- Abort
  - Modo de proteção para o acesso à memória
- Undefined

Modos de operação (Processor Modes)

- User
  - Modo de execução normal
- FIQ
  - Interrupções rápidas, com alta velocidade
- IRQ
  - Interrupções gerais
- Supervisor
  - Modo para o Sistema Operacional
- Abort
  - Modo de proteção para o acesso à memória
- Undefined
  - Instruções indefinidas, usadas na emulação de hardware

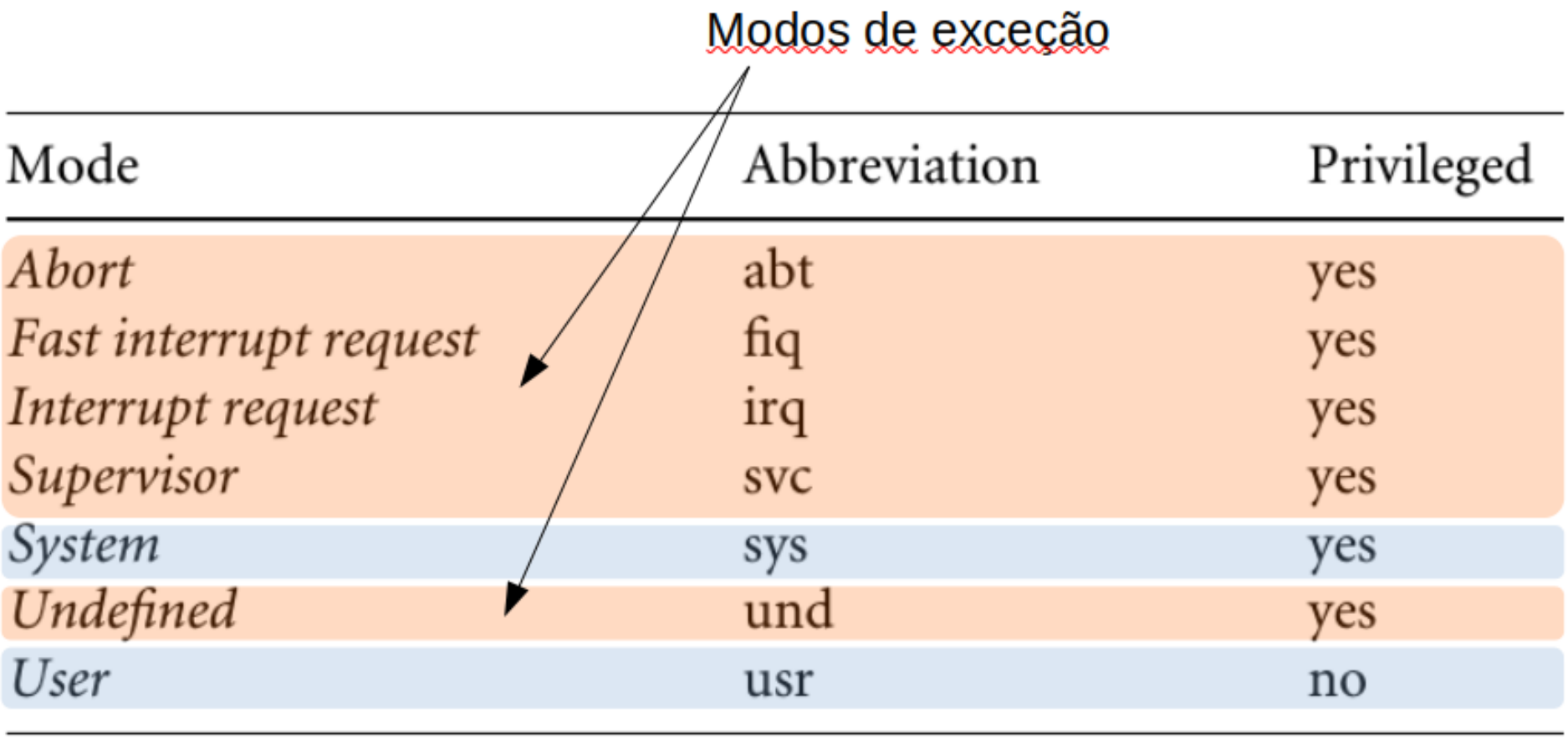
Modos de operação (Processor Modes)

- User
  - Modo de execução normal
- FIQ
  - Interrupções rápidas, com alta velocidade
- IRQ
  - Interrupções gerais
- Supervisor
  - Modo para o Sistema Operacional
- Abort
  - Modo de proteção para o acesso à memória
- Undefined
  - Instruções indefinidas, usadas na emulação de hardware
- System

Modos de operação (Processor Modes)

- User
  - Modo de execução normal
- FIQ
  - Interrupções rápidas, com alta velocidade
- IRQ
  - Interrupções gerais
- Supervisor
  - Modo para o Sistema Operacional
- Abort
  - Modo de proteção para o acesso à memória
- Undefined
  - Instruções indefinidas, usadas na emulação de hardware
- System
  - Modo de usuário com privilégios adicionais

Modos de operação (Processor Modes)

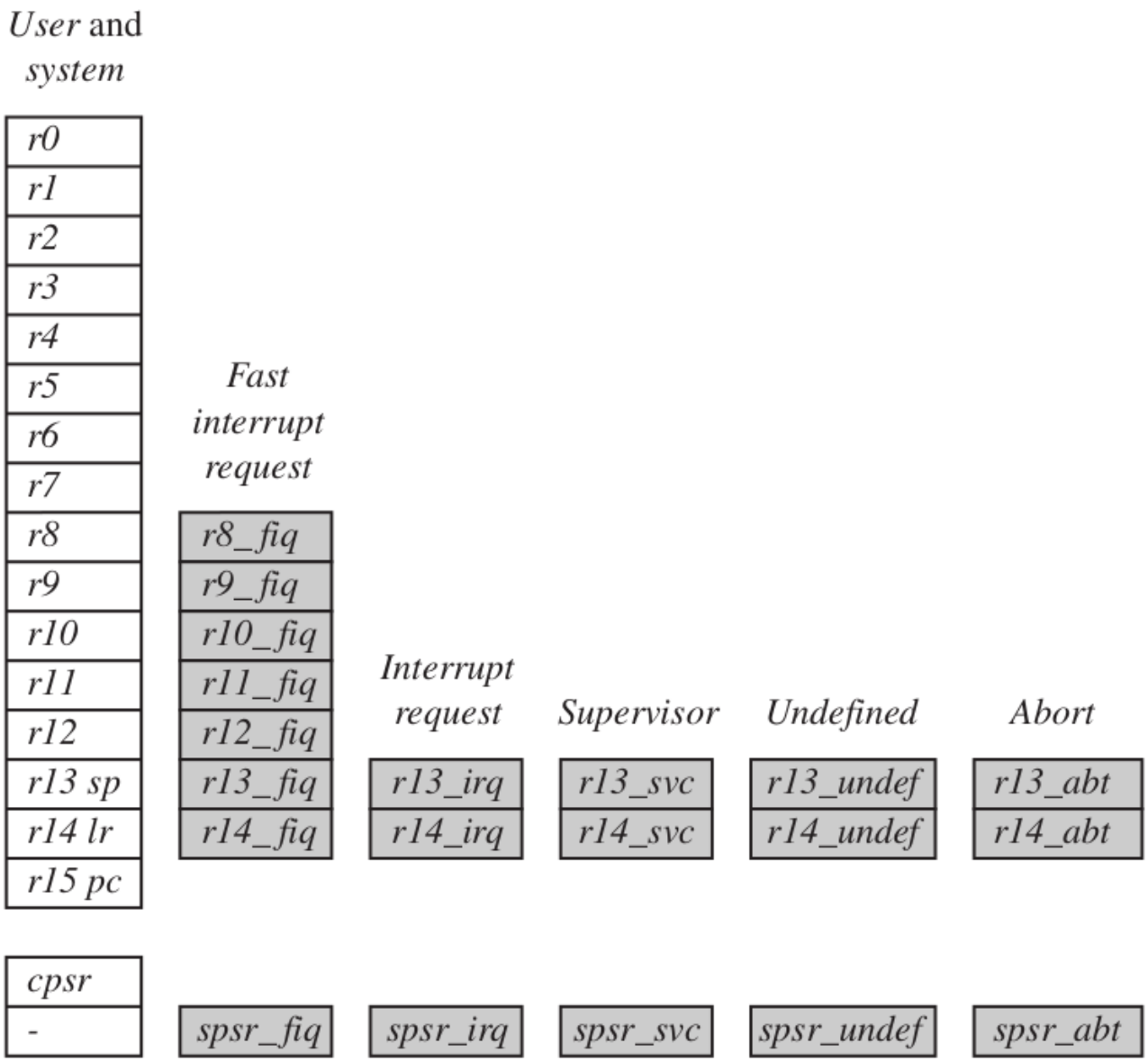


Mode	Abbreviation	Privileged
Abort	abt	yes
Fast interrupt request	fiq	yes
Interrupt request	irq	yes
Supervisor	svc	yes
System	sys	yes
Undefined	und	yes
User	usr	no

Modos de operação (Processor Modes)

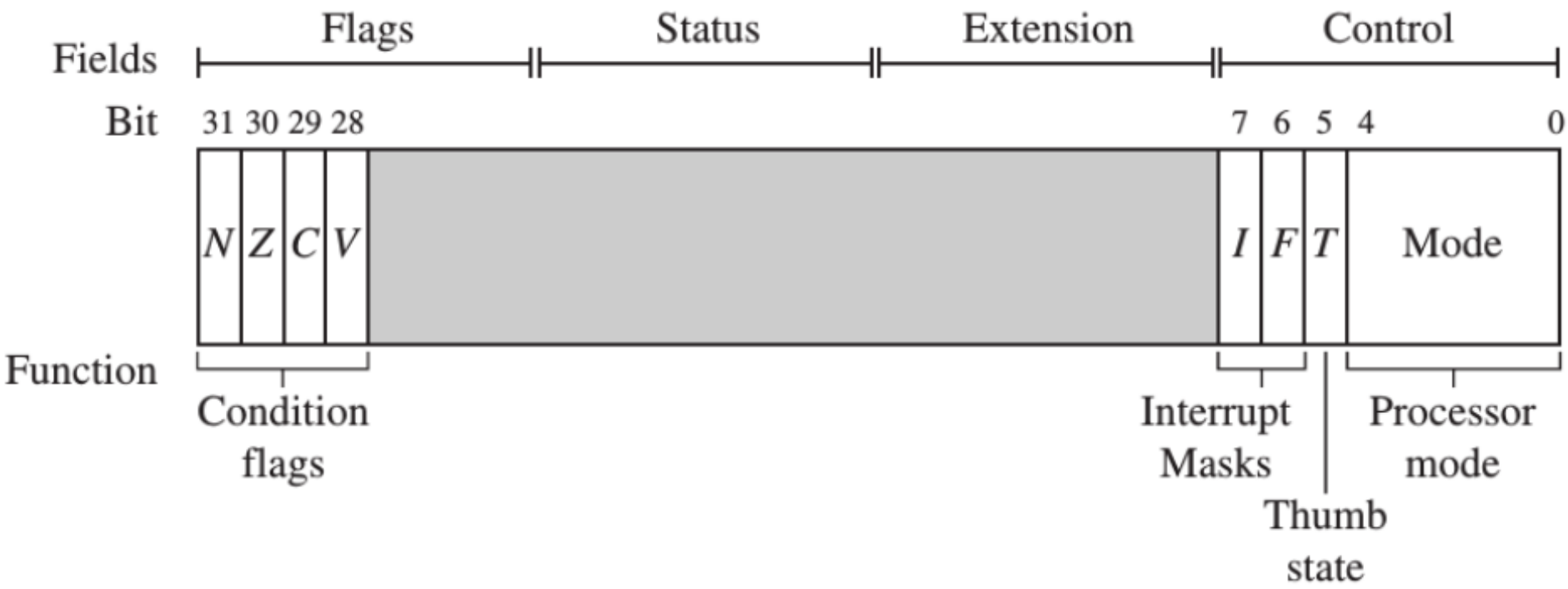
Mode	Abbreviation	Privileged	Mode[4:0]
Abort	abt	yes	10111
Fast interrupt request	fiq	yes	10001
Interrupt request	irq	yes	10010
Supervisor	svc	yes	10011
System	sys	yes	11111
Undefined	und	yes	11011
User	usr	no	10000

Registradores



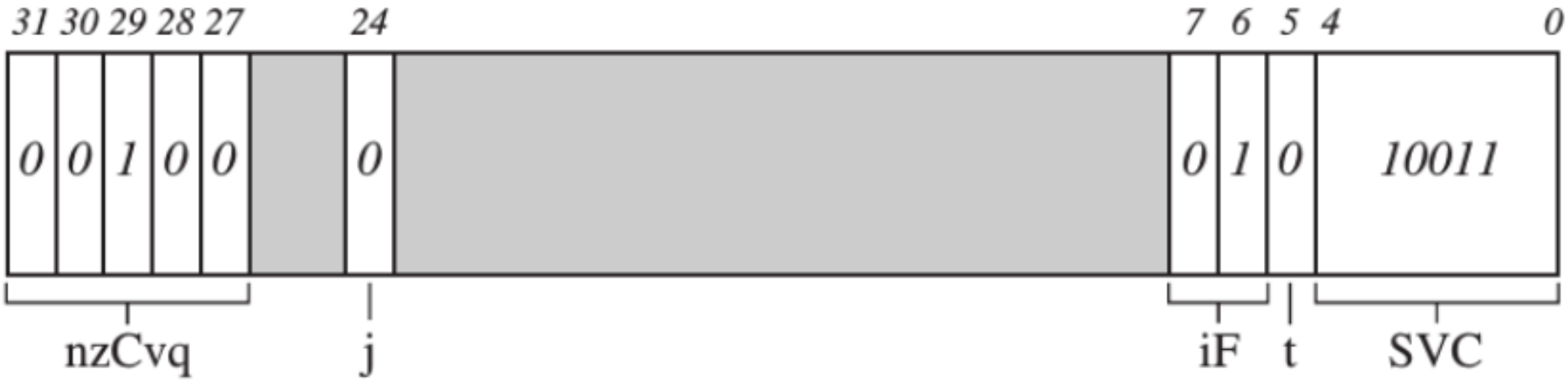
Modes						
	Privileged modes					
		Exception modes				
User	**System**	**Supervisor**	**Abort**	**Undefined**	**Interrupt**	**Fast interrupt**
R0	R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7	R7
R8	R8	R8	R8	R8	R8	R8_fiq
R9	R9	R9	R9	R9	R9	R9_fiq
R10	R10	R10	R10	R10	R10	R10_fiq
R11	R11	R11	R11	R11	R11	R11_fiq
R12	R12	R12	R12	R12	R12	R12_fiq
R13	R13	R13_svc	R13_abt	R13_und	R13_irq	R13_fiq
R14	R14	R14_svc	R14_abt	R14_und	R14_irq	R14_fiq
PC	PC	PC	PC	PC	PC	PC
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
		SPSR_svc	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq

Current Program Status Register (CPSR)



Current Program Status Register (CPSR)

Exemplo:
Carry Flag $\rightarrow$ C = 1
Modo ARM $\rightarrow$ T = 0 e J = 0
FIQ Desabilitada $\rightarrow$ F = 1
IRQ Habilitada $\rightarrow$ I = 0
Modo Supervisor $\rightarrow$ M = 10011



Mnemônicos de condição

Mnemonic	Name	Condition flags
EQ	equal	Z
NE	not equal	z
CS HS	carry set/unsigned higher or same	C
CC LO	carry clear/unsigned lower	c
MI	minus/negative	N
PL	plus/positive or zero	n
VS	overflow	V
VC	no overflow	v
HI	unsigned higher	zC
LS	unsigned lower or same	Z or c
GE	signed greater than or equal	NV or nv
LT	signed less than	Nv or nV
GT	signed greater than	NzV or nzv
LE	signed less than or equal	Z or Nv or nV
AL	always (unconditional)	ignored

Mnemônicos de condição – Exemplo

If (r0 != 10)
{
r1 = r1 + r0 - r2
}

Mnemônicos de condição – Exemplo

If (r0 != 10)
{
r1 = r1 + r0 - r2
}

Sem condicional

```
CMP r0, #10
BEQ FUNC
ADD r1, r1, r0
SUB r1, r1, r2
FUNC: ...
```

Mnemônicos de condição – Exemplo

If (r0 != 10)
{
r1 = r1 + r0 - r2
}

Sem condicional

```
CMP r0, #10
BEQ FUNC
ADD r1, r1, r0
SUB r1, r1, r2
FUNC: ...
```

Com condicional

```
CMP r0, #10
ADDNE r1,r1,r0
SUBNE r1,r1,r2
```

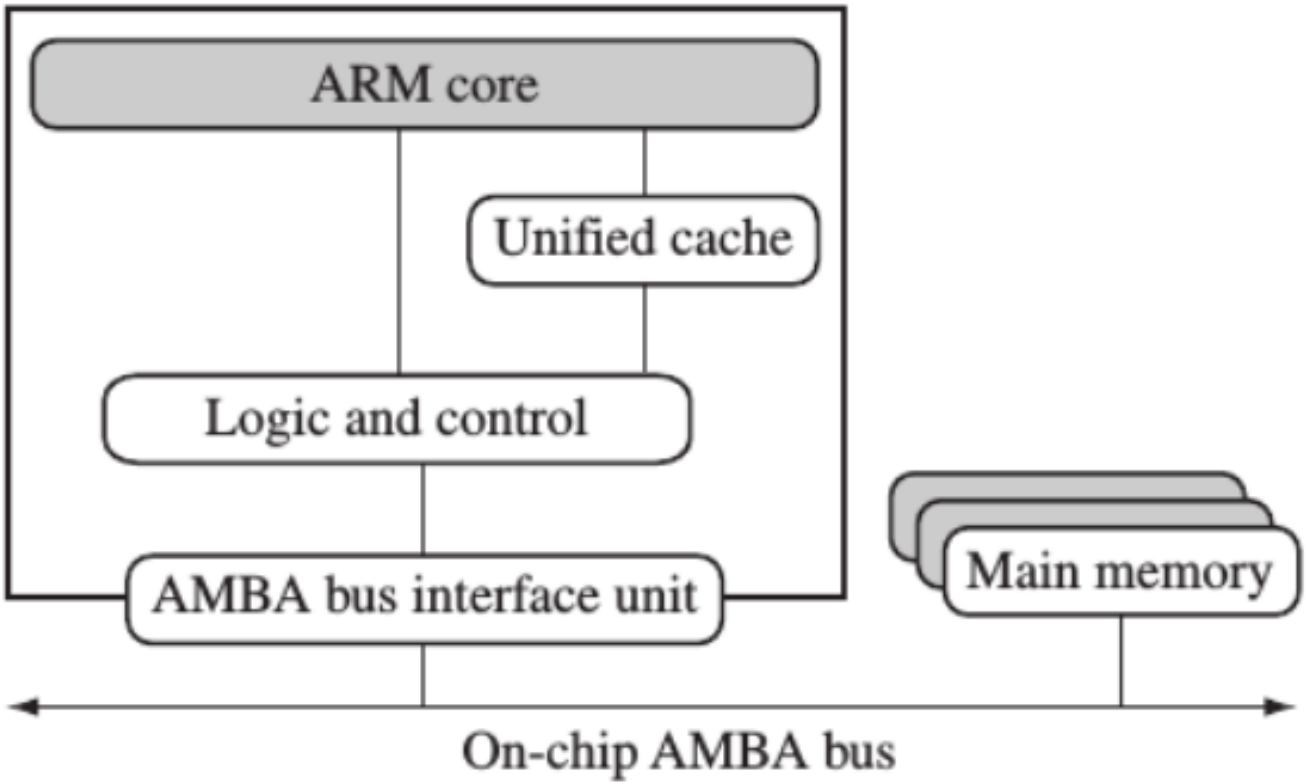
CPSR Mode bits

M[4:0]	Mode	Accessible registers
0b10000	User	PC, R14 to R0, CPSR
0b10001	FIQ	PC, R14_fiq to R8_fiq, R7 to R0, CPSR, SPSR_fiq
0b10010	IRQ	PC, R14_irq, R13_irq, R12 to R0, CPSR, SPSR_irq
0b10011	Supervisor	PC, R14_svc, R13_svc, R12 to R0, CPSR, SPSR_svc
0b10111	Abort	PC, R14_abt, R13_abt, R12 to R0, CPSR, SPSR_abt
0b11011	Undefined	PC, R14_und, R13_und, R12 to R0, CPSR, SPSR_und
0b11111	System	PC, R14 to R0, CPSR (ARM architecture v4 and above)

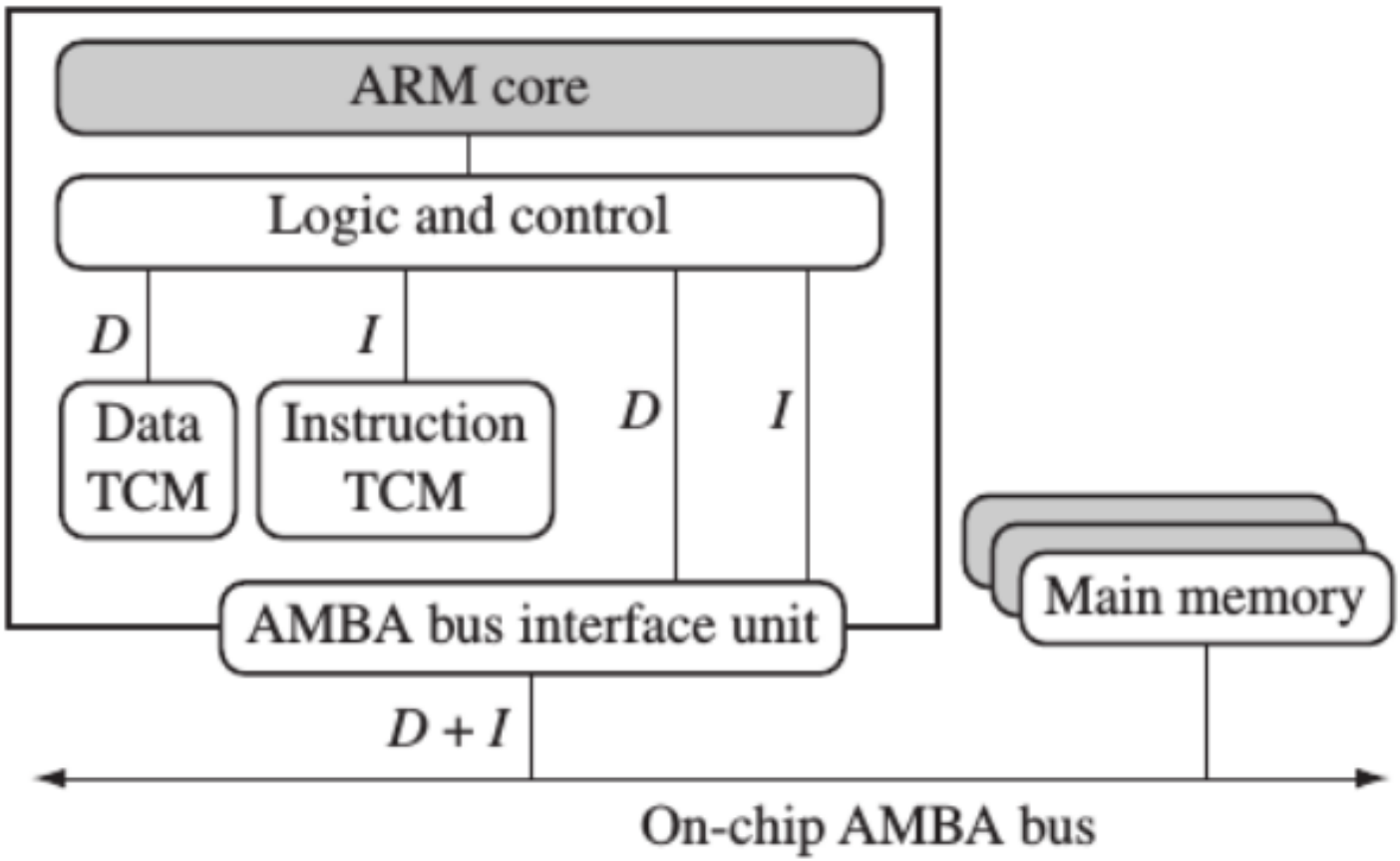
Conditional Flags

Flag	Flag name	Set when
Q	Saturation	the result causes an overflow and/or saturation
V	oVerflow	the result causes a signed overflow
C	Carry	the result causes an unsigned carry
Z	Zero	the result is zero, frequently used to indicate equality
N	Negative	bit 31 of the result is a binary 1

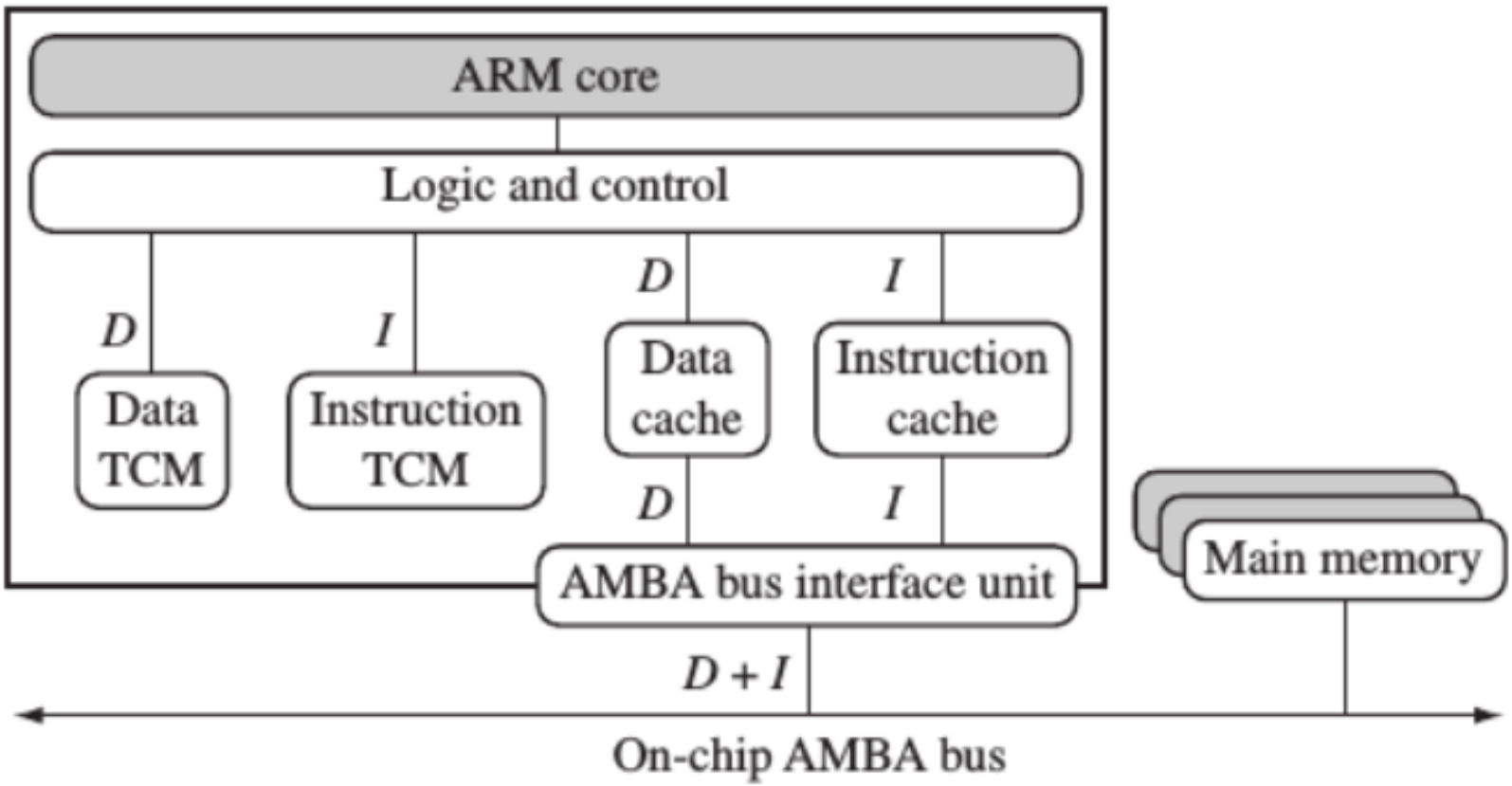
Cache - A simplified Von Neumann architecture with cache.



Tighly Coupled Memory (TCM) - A simplified Harvard architecture with TCMs.



Cache e TCM


ARM core
Logic and control
D
I
D
I
Data TCM
Instruction TCM
Data cache
Instruction cache
D
I
AMBA bus interface unit
Main memory
D + I
On-chip AMBA bus

Outras unidades

- MPU
- MMU
- CP15

Resumo de características do ISA

	ARM ($cpsr\ T = 0$)	Thumb ($cpsr\ T = 1$)
Instruction size	32-bit	16-bit
Core instructions	58	30
Conditional execution[a]	most	only branch instructions
Data processing instructions	access to barrel shifter and ALU	separate barrel shifter and ALU instructions
Program status register	read-write in privileged mode	no direct access
Register usage	15 general-purpose registers $+pc$	8 general-purpose registers +7 high registers $+pc$

	Jazelle ($cpsr\ T = 0, J = 1$)
Instruction size	8-bit
Core instructions	Over 60% of the Java bytecodes are implemented in hardware; the rest of the codes are implemented in software.

Nomenclatura

ARM{x}{y}{z}{T}{D}{M}{I}{E}{J}{F}{-S}

x—family
y—memory management/protection unit
z—cache
T—Thumb 16-bit decoder
D—JTAG debug
M—fast multiplier
I—EmbeddedICE macrocell
E—enhanced instructions (assumes TDMI)
J—Jazelle
F—vector floating-point unit
S—synthesizible version

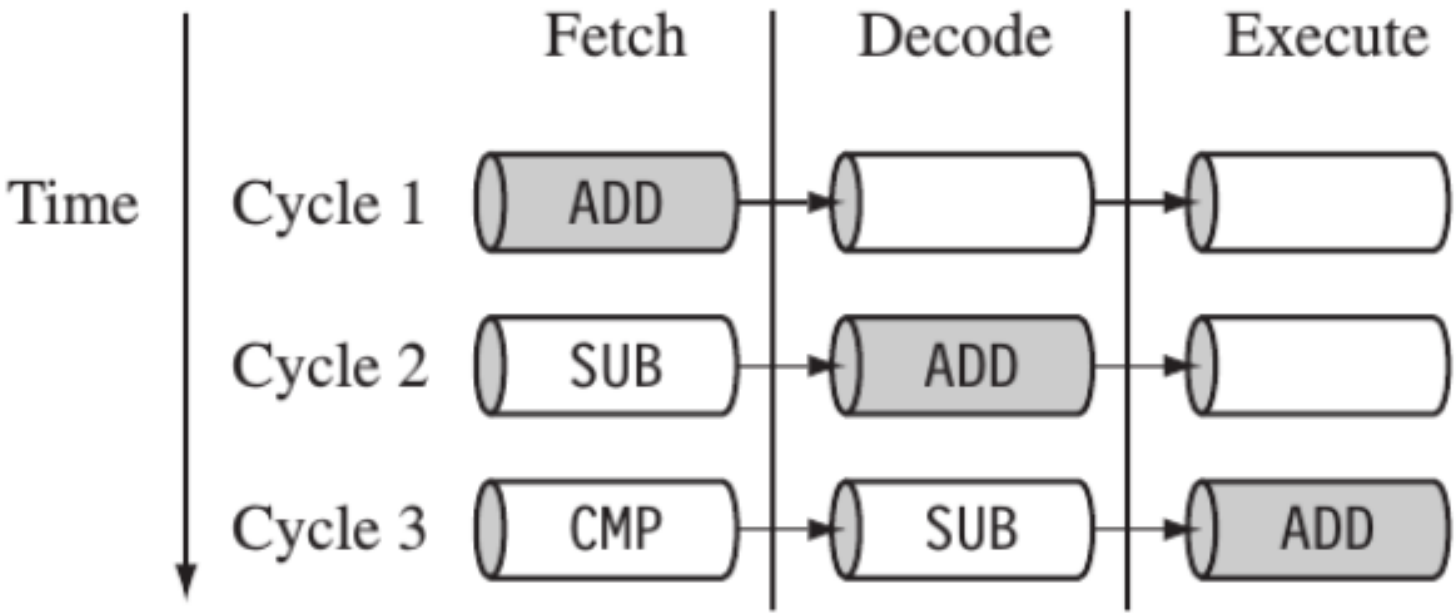
Comparação entre famílias ARM

	ARM7	ARM9	ARM10	ARM11
Pipeline depth	three-stage	five-stage	six-stage	eight-stage
Typical MHz	80	150	260	335
mW/MHz[a]	0.06 mW/MHz	0.19 mW/MHz (+ cache)	0.5 mW/MHz (+ cache)	0.4 mW/MHz (+ cache)
MIPS[b]/MHz	0.97	1.1	1.3	1.2
Architecture	Von Neumann	Harvard	Harvard	Harvard
Multiplier	8 × 32	8 × 32	16 × 32	16 × 32

[a] Watts/MHz on the same 0.13 micron process.

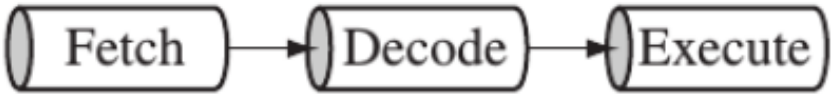
[b] MIPS are Dhrystone VAX MIPS.

Pipeline

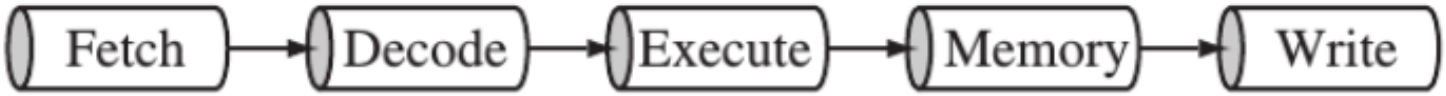


Pipelined instruction sequence.

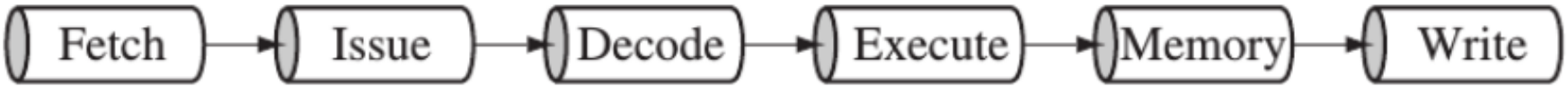
Pipeline



ARM7 Three-stage pipeline.

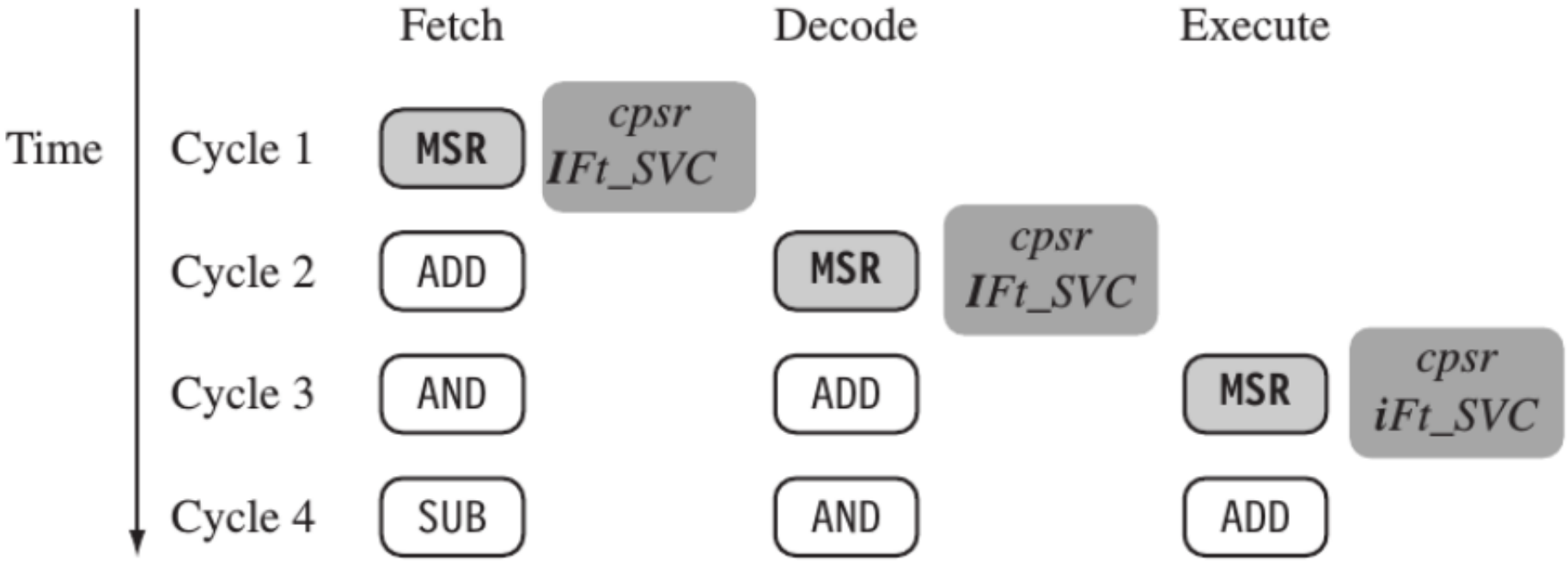


ARM9 five-stage pipeline.



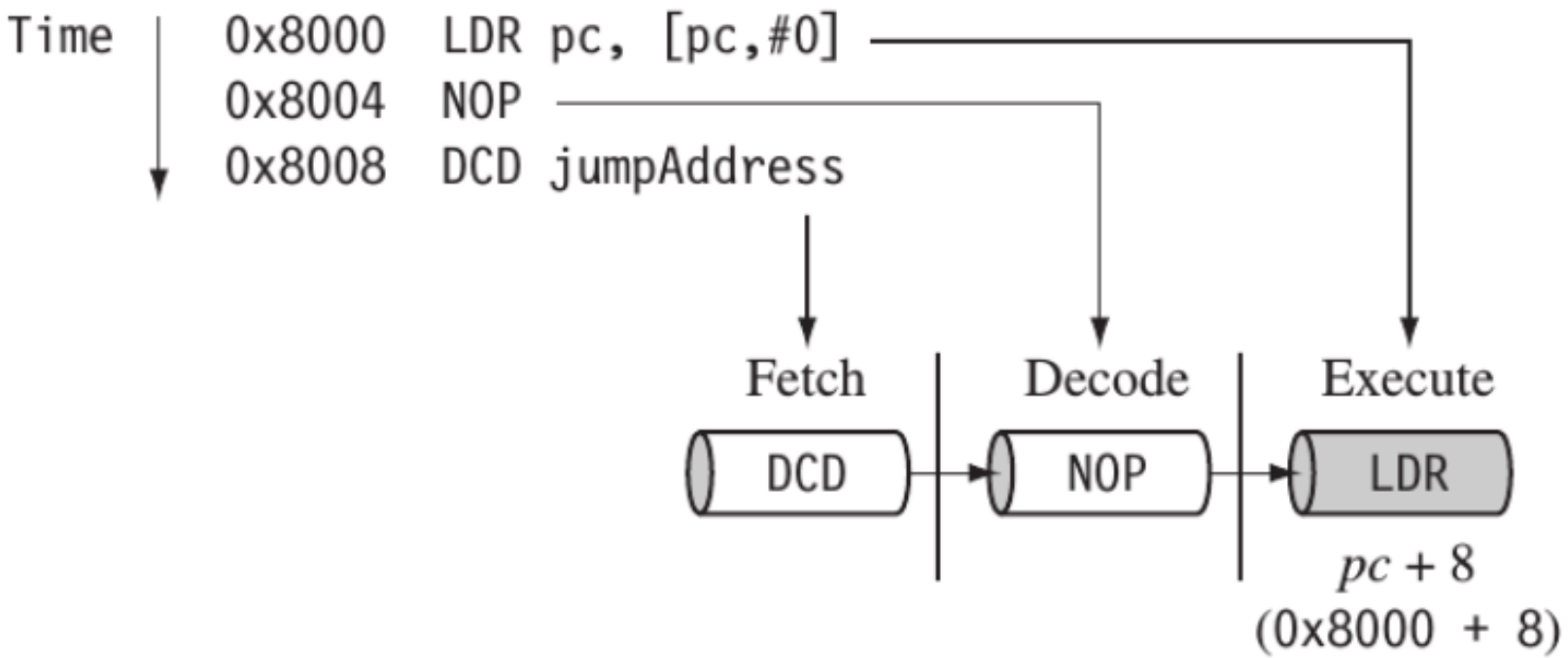
ARM10 six-stage pipeline.

Instruction Sequence



ARM instruction sequence.

Instruction Sequence



Example: $pc = \text{address} + 8$.

Tabela de vetores

- Quando uma interrupção ou exceção ocorre, o PC é setado para um endereço específico de uma região chamada de tabela de vetores.
  - O processador suspende a execução normal e carrega instruções da tabela de vetores

Tabela de vetores

- Quando uma interrupção ou exceção ocorre, o PC é setado para um endereço específico de uma região chamada de tabela de vetores.
  - O processador suspende a execução normal e carrega instruções da tabela de vetores

Exceção	Condição
Reset	→ Após reset
Undefined Instruction	→ Quando uma instrução não pode ser decodificada
Software Interrupt	→ Interrupções por software (instrução SWI)
Prefetch abort	→ Não há permissão para acessar uma instrução
Data abort	→ Não há permissão para acessar um dado
Interrupt request	→ Interrupção externa
Fast interrupt request	→ Interrupção externa de alta velocidade/prioridade

Tabela de vetores

Exception/interrupt	Shorthand	Address	High address
Reset	RESET	0x00000000	0xffff0000
Undefined instruction	UNDEF	0x00000004	0xffff0004
Software interrupt	SWI	0x00000008	0xffff0008
Prefetch abort	PABT	0x0000000c	0xffff000c
Data abort	DABT	0x00000010	0xffff0010
Reserved	—	0x00000014	0xffff0014
Interrupt request	IRQ	0x00000018	0xffff0018
Fast interrupt request	FIQ	0x0000001c	0xffff001c

Exceções

- Reset

```
R14_svc    = UNPREDICTABLE value
SPSR_svc   = UNPREDICTABLE value
CPSR[4:0]  = 0b10011                /* Enter Supervisor mode */
CPSR[5]    = 0                      /* Execute in ARM state */
CPSR[6]    = 1                      /* Disable fast interrupts */
CPSR[7]    = 1                      /* Disable normal interrupts */
if high vectors configured then
    PC     = 0xFFFF0000
else
    PC     = 0x00000000
```

Exceções

- Undefined

```
R14_und   = address of next instruction after the undefined instruction
SPSR_und  = CPSR
CPSR[4:0] = 0b11011              /* Enter Undefined mode */
CPSR[5]   = 0                    /* Execute in ARM state */
/* CPSR[6] is unchanged */
CPSR[7]   = 1                    /* Disable normal interrupts */
if high vectors configured then
    PC    = 0xFFFF0004
else
    PC    = 0x00000004
```

```
MOVS PC,R14
```

Exceções

- Software Interruption

```
R14_svc    = address of next instruction after the SWI instruction
SPSR_svc   = CPSR
CPSR[4:0]  = 0b10011                /* Enter Supervisor mode */
CPSR[5]    = 0                      /* Execute in ARM state */
/* CPSR[6] is unchanged */
CPSR[7]    = 1                      /* Disable normal interrupts */
if high vectors configured then
    PC     = 0xFFFF0008
else
    PC     = 0x00000008
```

```
MOVS PC,R14
```

Exceções

- Prefetch Abort

```
R14_abt    = address of the aborted instruction + 4
SPSR_abt   = CPSR
CPSR[4:0]  = 0b10111                  /* Enter Abort mode */
CPSR[5]    = 0                        /* Execute in ARM state */
/* CPSR[6] is unchanged */
CPSR[7]    = 1                        /* Disable normal interrupts */
if high vectors configured then
    PC     = 0xFFFF000C
else
    PC     = 0x0000000C
```

```
SUBS PC,R14,#4
```

Exceções

- Data Abort

```
R14_abt    = address of the aborted instruction + 8
SPSR_abt   = CPSR
CPSR[4:0]  = 0b10111                  /* Enter Abort mode */
CPSR[5]    = 0                        /* Execute in ARM state */
/* CPSR[6] is unchanged */
CPSR[7]    = 1                        /* Disable normal interrupts */
if high vectors configured then
    PC     = 0xFFFF0010
else
    PC     = 0x00000010
```

```
SUBS PC,R14,#8
```

OU

```
SUBS PC,R14,#4
```

Exceções

- IRQ

```
R14_irq   = address of next instruction to be executed + 4
SPSR_irq  = CPSR
CPSR[4:0] = 0b10010                 /* Enter IRQ mode */
CPSR[5]   = 0                       /* Execute in ARM state */
/* CPSR[6] is unchanged */
CPSR[7]   = 1                       /* Disable normal interrupts */
if high vectors configured then
    PC    = 0xFFFF0018
else
    PC    = 0x00000018
```

Exceções

- FIQ

```
R14_fiq   = address of next instruction to be executed + 4
SPSR_fiq  = CPSR
CPSR[4:0] = 0b10001                 /* Enter FIQ mode */
CPSR[5]   = 0                       /* Execute in ARM state */
CPSR[6]   = 1                       /* Disable fast interrupts */
CPSR[7]   = 1                       /* Disable normal interrupts */
if high vectors configured then
    PC    = 0xFFFF001C
else
    PC    = 0x0000001C
```

```
SUBS PC,R14,#4
```

Exceções

Priority		Exception
Highest	1	Reset
	2	Data Abort
	3	FIQ
	4	IRQ
	5	Prefetch Abort
Lowest	6	Undefined instruction SWI

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
```

QXD0133 - Arquitetura e Organização de Computadores II

Universidade Federal do Ceará - Campus Quixadá

Thiago Werlley
thiagowerlley@ufc.br

18 de outubro de 2025

Capítulo 2