

# CPU's



- Performance do CPU
- Consumo de energia/potência do CPU

15

15

## Elementos da performance do CPU



- Tempo de ciclo
- Pipeline do CPU
- Sistema de memória

16

16

## Pipelining

- Várias instruções são executadas simultaneamente em diferentes estágios do ciclo
- Várias condições podem causar *pipeline bubbles* que reduzem a utilização:
  - *branches* (saltos);
  - atrasos do sistema de memória;
  - etc.

17

17

## Medidas da Performance

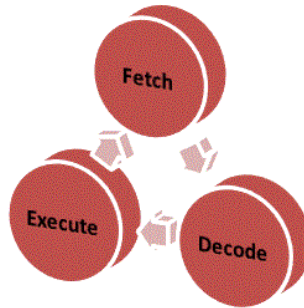
- **Latência**: tempo que uma instrução leva a entrar no *pipeline*.
- **Throughput**: número de instruções executadas por período de tempo
- O *pipelining* aumenta o *throughput* sem reduzir a latência

18

18

# Pipeline do ARM7

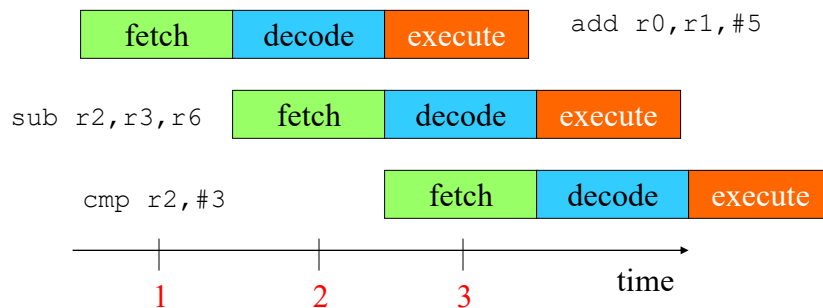
- O ARM 7 possui um *pipe* de 3 estágios:
  - *fetch* instrução da memória;
  - *decode* código da operação e operandos;
  - *execute*.



19

19

# Execução de pipeline no ARM



20

20

## Exemplo: tempo de execução do ARM

- Determinar o tempo de execução do filtro FIR (*Finite Impulse Response*):

```
for (i=0; i<N; i++)  
    f = f + c[i]*x[i];
```

- Apenas a instrução de *branch* (BLT) no teste do *loop* pode durar mais do que 1 ciclo
  - BLT `loop` ocupa 1 ciclo no melhor caso, 3 no pior caso

BLT – Branch on Lower Than

21

21

## Código do filtro FIR no ARM

### ; loop initiation code

```
MOV r0,#0 ; use r0 for i, set to 0  
MOV r8,#0 ; use a separate index for arrays  
ADR r2,N ; get address for N  
LDR r1,[r2] ; get value of N  
MOV r2,#0 ; use r2 for f, set to 0  
ADR r3,c ; load r3 with address of base of c  
ADR r5,x ; load r5 with address of base of x
```

### ; loop body

```
loop LDR r4,[r3,r8] ; get value of c[i]  
LDR r6,[r5,r8] ; get value of x[i]  
MUL r4,r4,r6 ; compute c[i]*x[i]  
ADD r2,r2,r4 ; add into running sum
```

### ; update loop counter and array index

```
ADD r8,r8,#4 ; add one to array index  
ADD r0,r0,#1 ; add 1 to i
```

### ; test for exit

```
CMP r0,r1  
BLT loop ;  
if i < N, continue loop  
loopend ...
```

22

22

# Performance do filtro FIR por bloco

Block	Variable	# instructions	# cycles
Initialization	$t_{init}$	7	7
Body	$t_{body}$	4	4
Update	$t_{update}$	2	2
Test	$t_{test}$	2	[2,4]

$$t_{loop} = t_{init} + N(t_{body} + t_{update}) + (N-1) t_{test, worst} + t_{test, best}$$

Loop no pior caso

Loop no melhor caso

## Cálculo:

Considerando  $N=10$

$$t_{loop} = 7 + 10(4 + 2) + (10-1)4 + 2$$

$$t_{loop} = 7 + 60 + 36 + 2$$

$$t_{loop} = 105 \text{ ciclos}$$

23

23

# Consumo de energia do CPU

- Os CPUs modernos são desenhados considerando o impacto do consumo de energia
- Potência vs. energia:
  - Calor depende do consumo de potência;
  - Tempo de vida da bateria depende do consumo de energia.

24

24

## Estratégias de redução de potência do CPU

- Reduzir a tensão da fonte
- Executar a uma frequência de relógio mais baixa;
- Desativar unidades de função com sinais de controlo, quando não estão em uso.
- Desligar componentes da fonte de alimentação quando não estão em uso

25

25

## Estilos de Gestão de Potência

- **Gestão de potência estática:** não depende da atividade do CPU
  - Exemplo: o utilizador ativa o modo de *power-down*
- **Gestão de potência dinâmica:** baseada na atividade do CPU
  - Exemplo: desativar unidades de funções

26

26

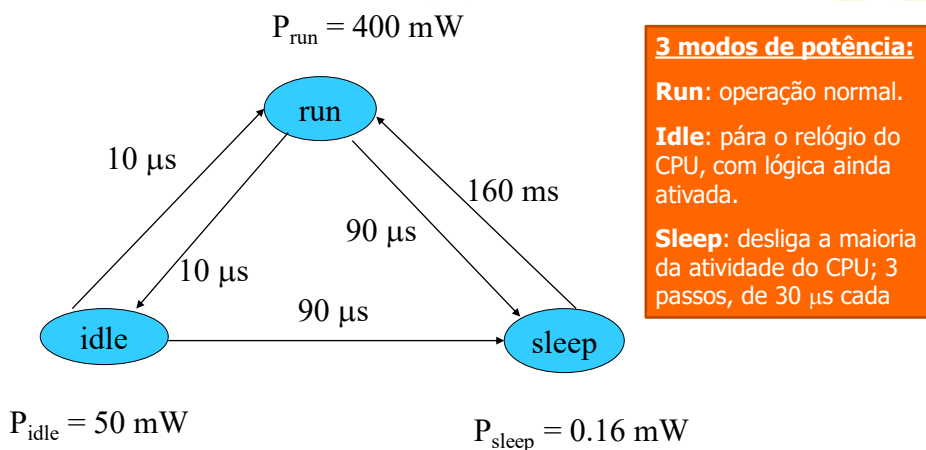
## Custos do modo *power-down*

- Ativar o modo *power-down* custa:
  - Tempo;
  - Energia.
- Deve determinar-se se o modo *power-down* é realmente vantajoso.
- Pode modelar-se os estados de potência do CPU com uma máquina de estados...

27

27

## Ex: Máquina de estados de potência do Strong Arm-1100



28

28

# Sistemas Embebidos

## Exercício 1

Determine o tempo médio de acesso à memória cache, com uma taxa de 35% .

$$t_{\text{cache}} = 5\text{ns}$$

$$t_{\text{main}} = 80\text{ns}$$

29

29

# Sistemas Embebidos

## Exercício 2

Determine o tempo médio de acesso à memória multi-nível, com um taxa de cache hit L1 de 25% e L2 35%:

$$t_{\text{cache L1}} = 2\text{ns}$$

$$t_{\text{cache L2}} = 10\text{ns}$$

$$t_{\text{main}} = 70\text{ns}$$

30

30



# Sistemas Embebidos

## Exercício 3

Determine o tempo médio de acesso à memória, onde num total de 500 acessos sequenciais ocorreram 100 cache miss:

$$t_{\text{cache}} = 4\text{ns}$$

$$t_{\text{main}} = 60\text{ns}$$

31

31

# Sistemas Embebidos

## Exercício 4

Determine o tempo médio de acesso à memória multi-nível, com uma taxa de cache hit L1 de 10% e 60% cache miss:

$$t_{\text{cache L1}} = 3\text{ ns}$$

$$t_{\text{cache L2}} = 20\text{ns}$$

$$t_{\text{main}} = 90\text{ns}$$

32

32

# Sistemas Embebidos

## Exercício 5

Determine a performance de um Filtro *FIR* de  $N=20$  numa *arquitetura ARM*, com o seguinte numero de instruções por bloco:

- Inicialização = 8
- Corpo = 5
- Atualização = 2
- Teste = 2-4

33

33

# Sistemas Embebidos

## Resolução da FT4

34