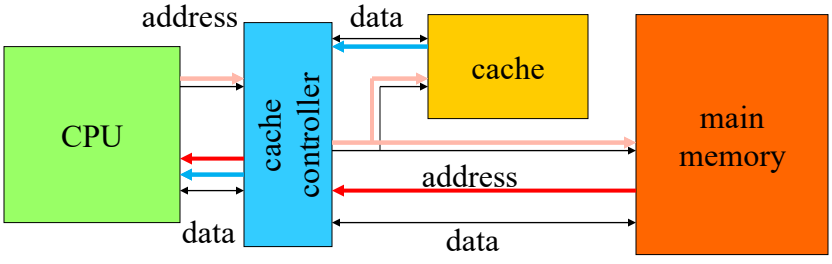
CPUs

- Caches
- Gestão de memória

Caches e CPUs

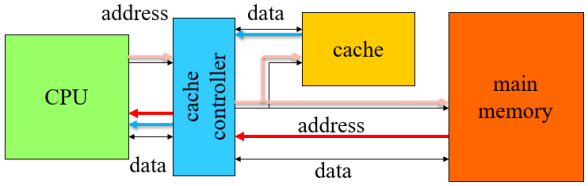


Operação de cache

- Um elevado número de localizações da memória principal estão mapeadas numa entrada de cache
- Há caches para:
  - instruções;
  - dados;
  - Dados + instruções (unificado).

Acesso à memória cache

- *Working set*: conjunto de localizações utilizadas por um programa num intervalo de tempo
- *Cache hit*: a localização requerida está na cache
- *Cache miss*: a localização requerida não está na cache



Tipo de “misses”

- Compulsório (*cold*): localização nunca foi acedida.
- Capacidade: *working set* é muito grande.
- Conflito: multiplas localizações no mapa *working set* para a mesma entrada na cache.

Performance do sistema de memória

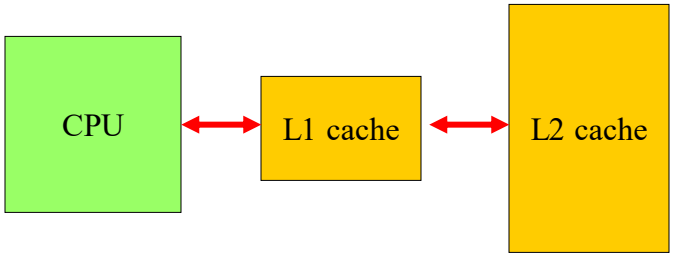
- h = taxa de *cache hit*.
- t_{cache} = tempo de acesso à cache
- t_{main} = Tempo de acesso à memória principal.
- Tempo médio de acesso à memória (t_{av}):

$$t_{av} = ht_{cache} + (1-h)t_{main}$$

Exemplo:
h = 0,60
t_{cache} = 2ns
t_{main} = 25ns

Cálculo:
t_{av}=0,6*2+(1-0,6)*25
t_{av}=11,2ns

Múltiplos níveis de cache



Tempo de acesso a caches multi-nível

- h_1 = taxa de *cache hit* em L1
- h_2 = taxa de *cache hit* em L2
- Tempo médio de acesso à memória:

$$t_{av} = h_1 t_{L1} + h_2 t_{L2} + (1- h_1 - h_2) t_{main}$$

Exemplo:
h1= 0,3
h2= 0,5
t_{cache} L1= 2ns
t_{cache} L2= 10ns
t_{main}= 40ns

Cálculo:
t_{av}=0,3*2+0,5*10 + (1-0,3-0,5)*40
t_{av}=0,6+5+8
t_{av}=13,6ns

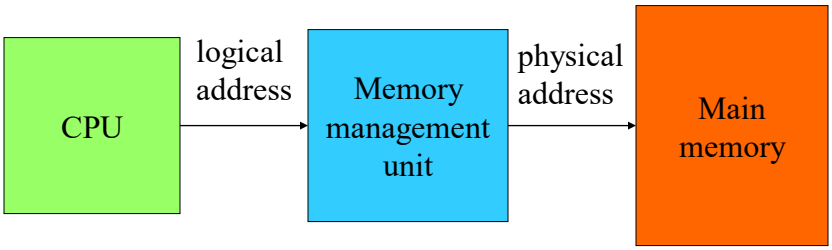
Benefícios da Performance da Cache

- Mantem na cache as localizações frequentemente acedidas
- A cache devolve mais de uma ***word*** em cada acesso
- Acessos sequenciais são mais rápidos após o primeiro acesso

Word/palavra: é uma unidade de dados com um comprimento definido (em bits) que pode ser endereçado e movido entre o sistema de memória e o processador.

Unidade de gestão de memória

- *Memory Management Unit (MMU)* traduz endereços:



MMU: Elemento de hardware que traduz os endereços entre o CPU e a memória física. az a correspondência de endereços lógicos passados do CPU para endereços físicos na memória.

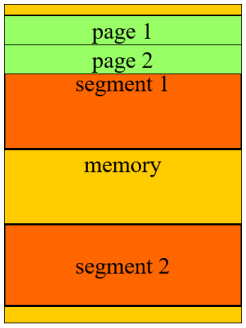
Tarefas de gestão de memória

- Permite que os programas se movam para a memória física durante a execução.
- Permite memória virtual:
  - Imagens da memória são mantidas na memória secundária
  - Imagens retornam à memória principal a pedido durante a execução
- *Page fault*: exceção gerada pela MMU, quando uma localização não está presente na memória principal.

Tradução de endereços

- Requer uma tabela para permitir mapeamentos arbitrários de endereços lógicos em físicos.
- Dois esquemas básicos são:
  - Segmentação *(segmented);*
  - Paginação *(paged).*
- Segmentação e paginação podem ser combinados (*x86*).

Tipos de região de memória: Segmentos e páginas



Paginação – Regiões de memoria divididas em páginas do mesmo tamanho com endereçamento (virtual) linear.

Segmentação – Os programas são divididos em segmentos/secções de tamanho variável cada um com o seu próprio espaço de endereçamento;

Gestão de Memória ARM

- Tipos de região de memória:
  - *section*: Bloco de 1 Mbyte;
  - *large page*: 64 kbytes;
  - *small page*: 4 kbytes.
- Um endereço é marcado numa *section-mapped ou page-mapped.*
- Esquema de tradução de 2 níveis (secção/ página)

CPUs

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

Elementos da performance do CPU

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

Pipelining

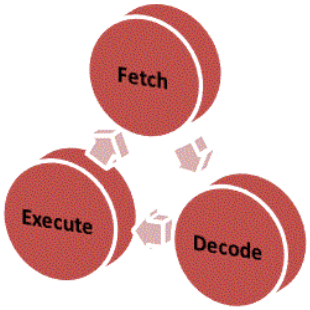
- Várias instruções são executadas simultaneamente em diferentes estágios do ciclo
- Várias condições podem causar *pipeline bubbles* que reduzem a utilização:
  - *branches* (saltos);
  - atrasos do sistema de memória;
  - etc.

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

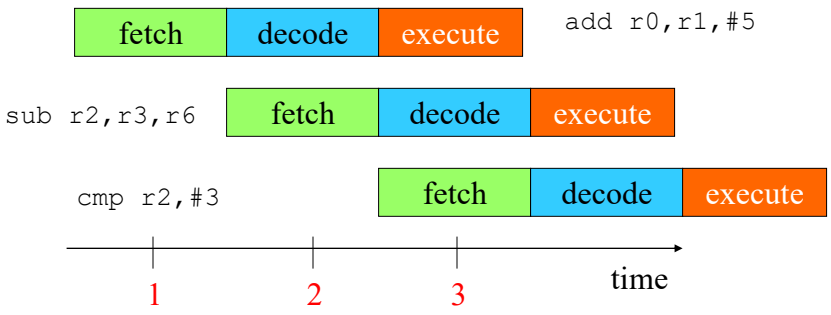
Pipeline do ARM7

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



19

Execução de pipeline no ARM



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

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

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 ciclos

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.

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

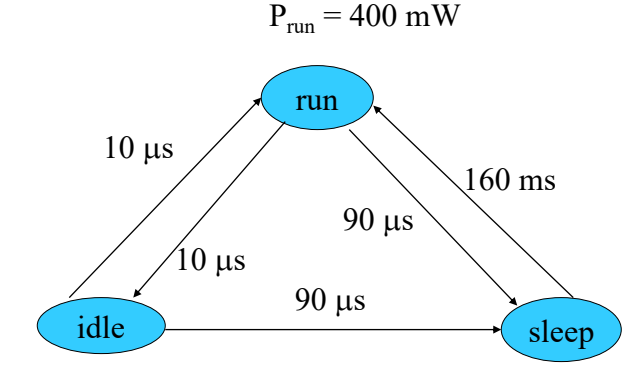
Estilos de Gestão de Potência

- 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

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…

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



3 modos de potência:

Run: operação normal.

Idle: pára o relógio do CPU, com lógica ainda ativada.

Sleep: desliga a maioria da atividade do CPU; 3 passos, de 30 µs cada

Sistemas Embebidos

Exercício 1

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

t_{cache}=5ns

t_{main}=80ns

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_{cache} L1=2ns

t_{cache} L2=10ns

t_{main}=70ns

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_{cache}=4ns

t_{main}=60ns

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_{cache} L1=3 ns

t_{cache} L2=20ns

t_{main}=90ns

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

Sistemas Embebidos

Resolução da FT4