Arquitectura de Computadores
Licenciatura em Engenharia Informática
Luís Paulo Peixoto dos Santos
Dep.de Informática - Universidade do Minho

Exercícios Propostos
Outubro, 2024

1 – Desempenho

1. Considere três processadores, CPU1, CPU2 e CPU3, que executam o mesmo *instruction set*, e que para um determinado programa P, apresentam as seguintes características:
   - CPU1 tem uma frequência de 3 GHz e um CPI = 1.5;
   - CPU2 tem uma frequência de 2.5 GHz e um CPI = 1.0;
   - CPU3 tem uma frequência de 4.0 GHz e um CPI = 2.2.

   a) Se o processador CPU1 executa o programa P em 10 segundos, qual o número de instruções deste programa?

   b) Indique, **justificando,** qual o processador que executa em menor tempo o programa P.

2. Justifique as três seguintes afirmações:

   a) "O aumento da frequência de um processador, sem qualquer outra alteração na sua organização, resulta num aumento do CPI".

   b) "Considere que um programa expresso numa linguagem de alto nível é compilado 2 vezes, com e sem optimização. Se a versão sem optimização (P-O0) implicar a execução de mais instruções do que a versão optimizada (P-O2) então, na generalidade dos casos, a versão P-O0 apresenta menor CPI do que P-O2"

   c) "Apesar de P-O0 apresentar menor CPI do que P-O2 é muito provável que o tempo de execução do primeiro seja maior do que o tempo de execução do segundo"

2 – Pipeline

1. Considere que a lógica combinatória dos diversos estágios do *datapath* estudado nas aulas (4 estágios: F – *fetch*; D – *decode*; E – *execute* ; W – *writeback*) têm as seguintes latências apresentadas na tabela abaixo. Os registos a utilizar têm uma latência de 50 ps.

F	D	E	W
200 ps	250 ps	150 ps	350 ps

   a) Indique, **justificando**, quais as máximas frequências para um processador sem *pipeline* e um processador com *pipeline*;

   b) Como relaciona o ganho de desempenho com o *pipelining* com o número máximo de estágios que podem ser usados? Quais os factores que determinam que o ganho possa ser diferente do número de estágios?

   c) A versão sem *pipeline* tem um CPI=1. Já na versão com *pipeline* e para um programa genérico, 12.5% das instruções resultam em dependências de dados ou controlo que exigem em média 2 ciclos adicionais. Qual o ganho obtido com a versão com *pipeline* para este programa?

2. Considere um processador com 5 estágios: F – *fetch*; D – *decode*; E1 – *execute1* ; E2 – *execute*2; W – *writeback*. Basicamente, a execução das instruções, associada à utilização da ALU ou de uma qualquer unidade funcional, está dividida em 2 estágios. O resultado a calcular pela unidade funcional só está disponível no final do estágio E2 (exemplo: se a instrução é `add %eax, %ebx` o resultado da adição só está disponível no fim de E2).
   Considere o programa:

```
add %esi, %edx
movl %edx, %eax
```

   a) Considere que este processador resolve as dependências de dados através do *stalling*, isto é, injecta `NOP`s no estágio E1, mantendo a instrução dependente no estágio de *Decode* até que a dependência esteja resolvida. Preencha a tabela abaixo, indicando para cada ciclo do relógio qual a instrução ou bolha (`NOP`) que se encontra em cada estágio.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10

b) Considere que este processador resolve as dependências de dados através de *data forwarding*, isto é, realimenta valores do estágio E2 para o estágio D e do registo WR (de suporte ao estágio W) para o estágio D. Injectará ainda assim NOPs no estágio E1, mantendo a instrução dependente no estágio de *Decode* até que o *data forwarding* seja possível ou a dependência esteja resolvida. Preencha a tabela abaixo, indicando para cada ciclo do relógio qual a instrução ou bolha (NOP) que se encontra em cada estágio.

	C1	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**	**C10**

3 – Hierarquia de Memória

1. A tabela abaixo apresenta o estado de uma cache com um total de 4 linhas, B=4 e m=5. A coluna L identifica a linha (em binário). A coluna *tag* apresenta o valor deste campo, com 3 bits, para as linhas cujo *valid* bit (coluna seguinte) esteja a 1. As 4 colunas seguintes apresentam, em hexadecimal, o valor de cada um dos *bytes* carregados na cache.

			Bytes			
L	**tag**	**valid**	**00**	**01**	**10**	**11**
00	100	1	0x23	0x7B	0xFF	0x00
01	--	0	--	--	--	--
10	001	1	0x0F	0xAC	0xCD	0x10
11	111	1	0x12	0x05	0x8F	0xD0

A próxima tabela apresenta o conteúdo da memória:

Addr	Val
00	0x23
01	0xBF
02	0XA0
03	0x05
04	0x0F
05	0xAC
06	0xCD
07	0x10

Addr	Val
08	0x01
09	0x02
10	0xCD
11	0xB2
12	0x02
13	0x23
14	0x9A
15	0xB4

Addr	Val
16	0x23
17	0x7B
18	0xFF
19	0x00
20	0xC1
21	0xD2
22	0xE3
23	0xB6

Addr	Val
24	0x22
25	0x55
26	0xFF
27	0x21
28	0x12
29	0x05
30	0x8F
31	0xD0

a) Indique qual a organização desta cache (S= , E= , B= , m=).

b) Considere um algoritmo de substituição LRU, durante a sequência de leitura de endereços de memória: 05, 03 e 26 (base 10). Para cada uma preencha os quadros, apenas com as alterações ao estado anterior. Indicando se trata de um ***hit*, *cold miss* ou colisão**, indicando **a linha** respetiva e ainda, o **conteúdo** da cache, sublinhando **valor do *byte* lido**, após cada acesso.

Endereço = 05				Bytes			
Hit/clM/coliS	*L*	*tag*	*valid*	**00**	**01**	**10**	**11**
Endereço = 03				Bytes			
Hit/clM/coliS	*L*	*tag*	*valid*	**00**	**01**	**10**	**11**
Endereço = 26				Bytes			
Hit/clM/coliS	*L*	*tag*	*valid*	**00**	**01**	**10**	**11**

2. Uma dada máquina tem uma frequência do relógio de 2GHz e a respectiva cache apresenta uma *miss rate* de instruções de 2% e de dados de 5%. A miss penalty é de 20 nanosegundos. O CPI do CPU é dado pela tabela abaixo para diferentes tipos de instruções:

Tipo instrução	CPI_{CPU}
Acesso memória	1
Restantes	2

Para um valor de %ecx = 10000 um dos excertos do programa abaixo executa em 75 microsegundos. Indique, justificando, qual.

```
excerto1:
  movl (%ebx), %eax
  addl %eax, %esi
  addl $4, %ebx
  decl %ecx
  jnz excerto1
```

```
excerto2:
     movl %ebx, %eax
     addl %eax, %esi
     addl $4, %ebx
     subl $2, %ecx
     jnz excerto2
```

3. Considere uma máquina com cache (S=1; E=4; B=8; m=8). Considere um programa que vai aceder a 8 elementos de um vector v, pela seguinte ordem: `v[0], v[8], v[1], v[9], v[2], v[10], v[3], v[11]`. Cada elemento do vector ocupa 4 *bytes*. O primeiro elemento do vector está guardado em memória no endereço 32, isto é, `&v[0]=32`.
   a) Qual a *miss rate* exibida nos acessos a v?
   b) Consegue propor uma ordem diferente de acesso aos mesmos elementos de v que resulte numa *miss rate* significativamente menor?
   c) Para a ordem original de acessos a v, qual seria a *miss rate* se a organização da cache fosse (S=2; E=2; B=8; m=8)?