

## Notas Importantes!

1. Verifique, para todas as questões, qual a resposta correcta e assinale com uma cruz a sua escolha na tabela ao lado. Por cada resposta incorrecta será descontada, à cotação global, 1/3 da cotação da respectiva pergunta.
2. Durante a realização do teste não é permitida a permanência na sala de calculadoras, telemóveis ou outros dispositivos electrónicos.

## Grupo I

- ✓ 1. Uma arquitectura do tipo *Harvard* é caracterizada por:
- a. ter segmentos de memória independentes para dados e para código.
  - b. ter dois barramentos de dados e um barramento de endereços.
  - c. partilhar a mesma memória entre dados e instruções.
  - d. permitir o acesso a instruções e dados no mesmo ciclo de relógio.
- ✓ 2. Quando um endereço se obtém da adição do conteúdo de um registo com um *offset* constante:
- a. diz-se que estamos perante um endereçamento imediato.
  - b. diz-se que estamos perante um endereçamento directo a registo com *offset*.
  - c. diz-se que estamos perante um endereçamento indirecto a registo com deslocamento.
  - d. diz-se que estamos perante um endereçamento indirecto relativo a PC.
- ✓ 3. Na convenção adoptada pela arquitectura MIPS, a realização de uma operação de *pop* da *stack* do valor do registo *\$ra* é realizada pela seguinte sequência de instruções:
- a. `addu $sp, $sp, 4` seguida de `lw $ra, 0($sp)`.
  - b. `lw $ra, 0($sp)` seguida de `addu $sp, $sp, 4`.
  - c. `lw $ra, 0($sp)` seguida de `subu $sp, $sp, 4`.
  - d. `subu $sp, $sp, 4` seguida de `lw $ra, 0($sp)`.
- ✓ 4. A detecção de *overflow* numa operação de adição de números com sinal faz-se através:
- a. do "ou" exclusivo entre o *carry in* e o *carry out* da célula de 1 bit mais significativa.
  - b. da avaliação do bit mais significativo do resultado.
  - c. do "ou" exclusivo entre os 2 bits mais significativos do resultado.
  - d. da avaliação do *carry out* do bit mais significativo do resultado.
- ✓ 5. Considerando que o código ASCII do carácter '0' é 0x30 e que os valores das três *words* armazenadas em memória a partir do endereço 0x10010000 são 0x30313200, 0x33343536 e 0x37380039, num computador MIPS *little endian* a *string* ASCII armazenada a partir do endereço 0x10010001 é:
- a. "21065439".
  - b. "65439".
  - c. "12".
  - d. "345678".
- ✓ 6. Considerando que *\$t0* = -4 e *\$t1* = 5, o resultado da instrução `mult $t0, $t1` é:
- a. *HI* = 0x80000000, *LO* = 0x000000EC.
  - b. *HI* = 0xFFFFFFFF, *LO* = 0xFFFFFFFFEC.
  - c. *HI* = 0xFFFFFEEC, *LO* = 0xFFFFFFFF.
  - d. *HI* = 0x00000000, *LO* = 0xFFFFFEEC.
- ✓ 7. A decomposição numa sequência de adições e subtracções, de acordo com o algoritmo de Booth, da quantidade binária 010110<sub>(2)</sub>:
- $$\begin{array}{ccccccc} & 5 & 4 & 3 & 2 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{array}$$
- $$\begin{array}{ccccccc} & + & - & + & 0 & - & \end{array}$$
- a.  $-2^1 + 2^3 - 2^4 + 2^5 + 2^6$
  - b.  $+2^0 - 2^2$
  - c.  $+2^1 - 2^3 + 2^4 - 2^5$
  - d.  $-2^0 + 2^2$
- ✓ 8. Os endereços mínimo e máximo para os quais uma instrução de salto incondicional ("j") da arquitectura MIPS, presente no endereço 0x0043FFFC pode saltar são:
- a. 0x0041FFFC, 0x0045FFF8.
  - b. 0x00420000, 0x0045FFFC.
  - c. 0x00000000, 0xFFFFFFFF.
  - d. 0x00000000, 0x0FFFFFFC.
- ? 9. No MIPS, as instruções do tipo "I" incluem um campo imediato de 16 bits que:
- a. permite armazenar um *offset* de endereçamento de  $\pm 32$  KBytes para as instruções "sw".
  - b. permite armazenar um *offset* de endereçamento de  $\pm (32 \cdot 4)$  KBytes para as instruções "sw".
  - c. permite armazenar um *offset* de endereçamento de  $\pm 32$  KBytes para as instruções "beq".
  - d. permite armazenar um *offset* de endereçamento de  $\pm (32 \cdot 4)$  Kilo instruções para as instruções.

	a	b	c	d
1	X			
2			X	
3		X		
4	X			
5		X		
6		X		
7	X			
8		X		
9	X			
10				X
11			X	
12		X		
13	X			
14				X
15				X
16	X			
17				X
18				X
19			X	
20				
21	X			
22		X		
23			X	
24		X		
25			X	
26			X	
27			X	
28				

11/5

3/4

2/1



10. Numa implementação single-cycle da arquitectura MIPS:
- existe uma única ALU para realizar todas as operações aritméticas e lógicas necessárias para executar num único ciclo de relógio qualquer uma das instruções suportadas.
  - existem registos à saída dos elementos operativos fundamentais para guardar valores a utilizar no ciclo de relógio seguinte.
  - todas as operações de leitura e escrita são síncronas com o sinal de relógio.
  - ☒ existem memórias específicas para código e dados para possibilitar o acesso a ambos os tipos de informação num único ciclo de relógio.
11. A frequência de relógio de uma implementação single cycle da arquitectura MIPS:
- é limitada pelo maior dos tempos de atraso dos elementos operativos Memória, ALU e File Register.
  - varia em função da instrução que está a ser executada.
  - ☒ é limitada pelo maior dos atrasos cumulativos dos elementos operativos envolvidos na execução da instrução mais longa.
  - é limitada pelo menor dos tempos de atraso dos elementos operativos Memória, ALU e File Register.
12. A unidade de controlo de uma implementação multi-cycle da arquitectura MIPS:
- é um elemento combinatório que gera os sinais de controlo em função do campo *opcode* do código máquina da instrução.
  - ☒ é uma máquina de estados em que o primeiro e o segundo estados são comuns à execução de todas as instruções.
  - é uma máquina de estados com um número de estados igual ao número de fases da instrução mais longa.
  - é um elemento combinatório que gera os sinais de controlo em função do campo *funct* do código máquina da instrução.
13. Numa implementação multi-cycle da arquitectura MIPS, na segunda e terceira fases de execução de uma instrução de salto condicional ("**beq/bne**"), a ALU é usada, pela ordem indicada, para:
- ☒ calcular o valor do *Branch Target Address* e comparar os registos (operandos da instrução).
  - calcular o valor de PC+4 e comparar os registos (operandos da instrução).
  - comparar os registos (operandos da instrução) e calcular o valor do *Branch Target Address*.
  - calcular o valor de PC+4 e o valor do *Branch Target Address*.
14. Uma implementação pipelined de uma arquitectura possui, relativamente a uma implementação single-cycle da mesma, a vantagem de:
- diminuir o tempo de execução de cada uma das instruções.
  - permitir a execução de uma nova instrução a cada novo ciclo de relógio.
  - aumentar o débito de execução das instruções.
  - ☒ todas as anteriores.
15. A frequência de relógio de uma implementação pipelined da arquitectura MIPS:
- é limitada pelo maior dos atrasos cumulativos dos elementos operativos envolvidos na execução da instrução mais longa.
  - é definida de forma a evitar *stalls*, assim como *delay slots*.
  - é limitada pelo menor dos tempos de atraso dos elementos operativos Memória, ALU e File Register.
  - ☒ é limitada pelo maior dos tempos de atraso dos elementos operativos Memória, ALU e File Register.
16. A técnica de *forwarding/bypassing* num processador MIPS pipelined permite:
- ☒ utilizar como operando de uma instrução um resultado produzido por outra instrução que se encontra numa etapa mais recuada do *pipeline*.
  - trocar a ordem de execução das instruções de forma a resolver um *hazard* de dados.
  - utilizar como operando de uma instrução um resultado produzido por outra instrução que se encontra numa etapa mais avançada do *pipeline*.
  - escrever o resultado de uma instrução no *File Register* antes de ela chegar à etapa WB.

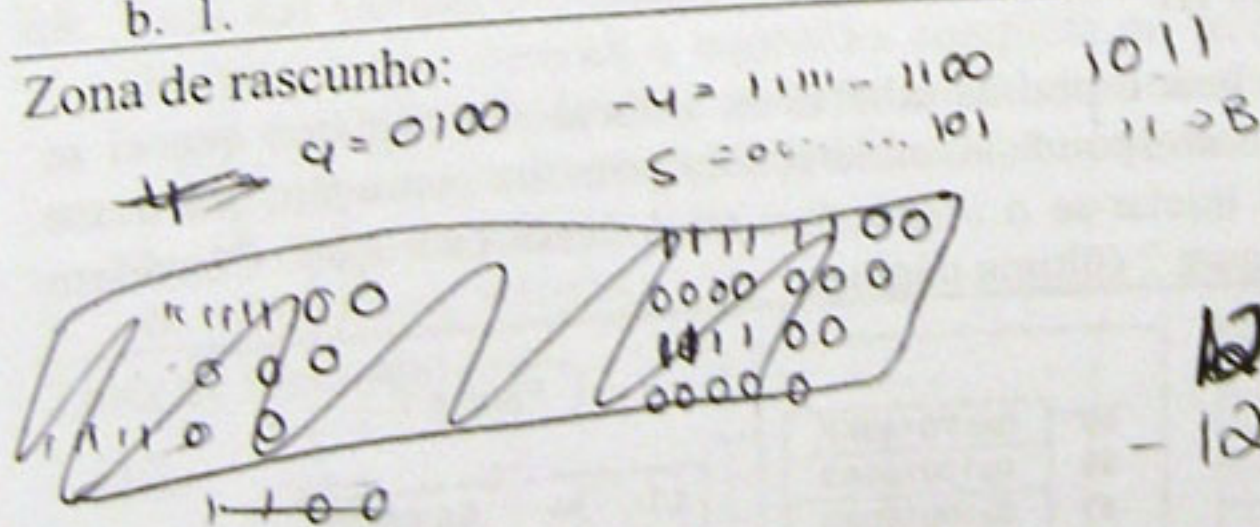
Zona de rascunho:



## Grupo II

17. O código máquina da instrução `sw $3, -128($4)`, representado em hexadecimal, é (considerando que para esta instrução `opcode=0x2B`):
- `0xAC838080`.
  - `0xAC648080`.
  - `0xAC64FF80`.
  - `0xAC83FF80`.
18. Considere que `a=0xC0D00000` representa uma quantidade codificada em hexadecimal segundo a norma IEEE 754 precisão simples. O valor representado em "a" é, em notação decimal:
- $-3,25 \times 2^1$ .
  - $-0,1625 \times 2^3$ .
  - $-0,1625 \times 2^1$ .
  - $-16,25 \times 2^1$ .
19. Considerando que `$f2=0x3A600000` e `$f4=0xBA600000`, o resultado da instrução `sub.s $f0, $f2, $f4` é:
- `$f0=0x39E00000`.
  - `$f0=0x80000000`.
  - `$f0=0x00000000`.
  - `$f0=0x3AE00000`.
20. Numa implementação *single cycle* da arquitectura MIPS, a frequência máxima de operação imposta pela instrução de leitura da memória de dados é, assumindo os atrasos a seguir indicados:
- Memórias externas: leitura - 9ns, escrita - 11ns; File register: leitura - 3ns, Escrita - 4ns;  
Unidade de Controlo: 2ns; ALU (qualquer operação): 7ns; Somadores: 4ns; Outros: 0ns.
- 32,25 MHz (T=31ns).
  - 31,25 MHz (T=32ns).
  - 29,41 MHz (T=34ns).
  - 25,00 MHz (T=40ns).
21. Considerando as seguintes frequências relativas de instruções de um programa a executar num processador MIPS: `lw` - 20%; `sw` - 10%; `tipo R` - 50%; `beq/bne` - 15%; `j` - 5%, a melhoria de desempenho proporcionada por uma implementação *multi-cycle* a operar a 100 MHz relativamente a uma *single-cycle* a operar a 20 MHz é de:
- 5.
  - 1.
  - 1,25.
  - 0,8.

Zona de rascunho:



Handwritten calculations:

$$128 - 128 = 0$$

$$10000000$$

$$00000000 \quad 10000000$$

$$11111111 \quad 10000000$$

$$FF \quad 80$$

Handwritten calculations and diagrams:

- OP:  $1010101100100011$
- $5AC83FF80$  (circled)
- $64 - 52 = 116$
- $116 - 127 = -11$
- $4 = -11$
- $\$f0 = 0$
- $129 - 127 = 2$
- $0,5$
- $125 = 0,625$
- $-1,625 \times 2^2 = -6 \times 25 \times 2^1$  (circled)
- $64 \times 2 = 128$
- $128 \times 3$
- $30313200$
- $2048 \times 128 = 262144$
- $262144 \times 16$
- $4194304$
- $29$
- $PC+4$
- $4ms$
- $3ms$
- $4ms$
- $4ms$



22. Um *hazard* de controlo numa implementação *pipelined* de um processador ocorre quando:
- um dado recurso de hardware é necessário para realizar no mesmo ciclo de relógio duas ou mais operações relativas a instruções em diferentes etapas do *pipeline*.
  - é necessário fazer o *instruction fetch* de uma nova instrução e existe numa etapa mais avançada do *pipeline* uma instrução que ainda não terminou e que pode alterar o fluxo de execução.
  - existe uma dependência entre o resultado calculado por uma instrução e o operando usado por outra que segue mais atrás no *pipeline*.
  - por azar, a unidade de controlo desconhece o *opcode* da instrução que se encontra na etapa ID.
23. O seguinte trecho de código, a executar sobre uma implementação *pipelined* da arquitectura MIPS, apresenta os seguintes *hazards*:
- um *hazard* de controlo na quarta instrução e um *hazard* de dados na segunda instrução que pode ser resolvido por *forwarding*.
  - um *hazard* estrutural na primeira instrução e um *hazard* de controlo na quarta instrução.
  - um *hazard* de controlo na quarta instrução e *hazards* de dados na segunda, terceira e na quarta instruções que podem ser resolvidos por *forwarding*.
  - um *hazard* de controlo na quarta instrução e *hazards* de dados na terceira e na quarta instruções que podem ser resolvidos por *forwarding*.
- ```

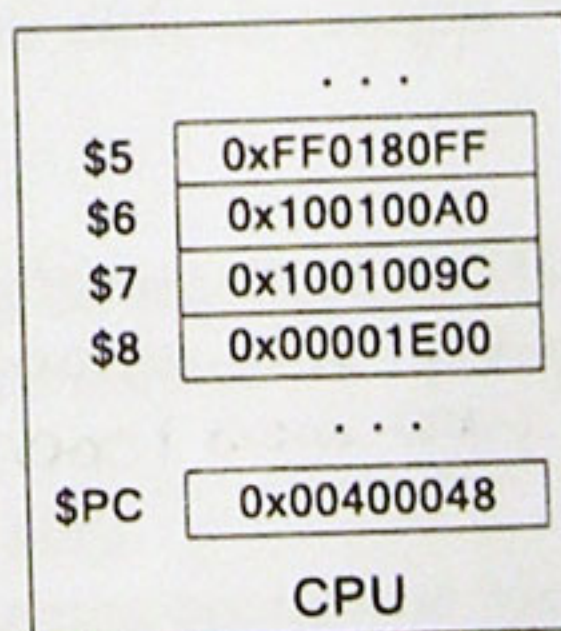
L1: lw    $t0, 0($t1)    # 1
      add  $t2, $t3, $t4  # 2
      or   $t1, $t2, $t0  # 3
      beq  $t5, $t1, L1   # 4
  
```
24. Considere o *datapath* e a unidade de controlo fornecidos na figura da última página (com ligeiras alterações relativamente à versão das aulas teórico-práticas) correspondendo a uma implementação *multi-cycle* simplificada da arquitectura MIPS. Admita que os valores indicados no *datapath* fornecido correspondem à "fotografia" tirada no decurso da execução de uma instrução. Tendo em conta todos os sinais, pode-se concluir que está em execução a instrução:
- `lw $6, 0x2020($5)` na terceira fase.
  - `add $4, $5, $6` na quarta fase.
  - `add $4, $5, $6` na terceira fase.
  - `lw $6, 0x2020($5)` na quinta fase.

## Grupo III

Considere o trecho de código apresentado na Figura 1, bem como as tabelas os valores dos registos que aí se apresentam. Admita que o valor presente no registo **\$PC** corresponde ao endereço da primeira instrução, que nesse instante o conteúdo dos registos é o indicado, e que vai iniciar-se o *instruction fetch* dessa instrução. Considere ainda o *datapath* e a unidade de controlo fornecidos na Figura 2 (última página).

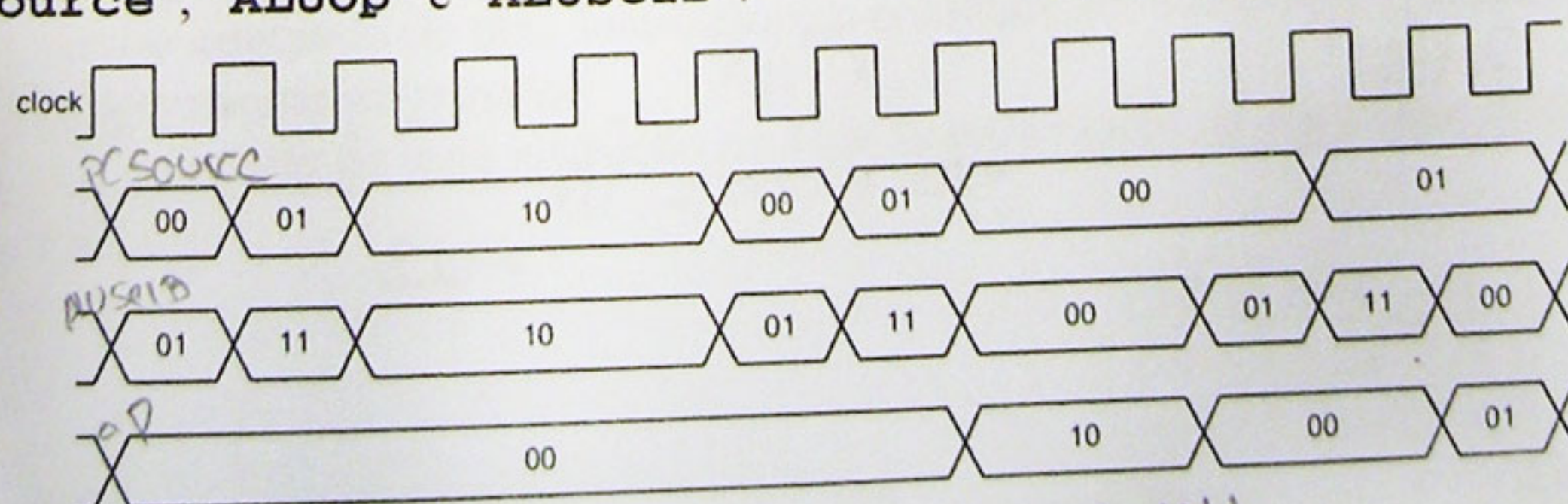
| Endereço   | Dados      |
|------------|------------|
| 0x1001009C | 0xFFFF0000 |
| 0x100100A0 | 0x021B581A |
| 0x100100A4 | 0x00008000 |
| 0x100100A8 | 0x1B54E790 |
| 0x100100AC | 0x00FE7F00 |
| 0x100100B0 | 0x5FF38C29 |
| ...        | ...        |

| Opcode | Funcnt | Operação |
|--------|--------|----------|
| 0      | 0x20   | add      |
| 0      | 0x22   | sub      |
| 0      | 0x24   | and      |
| 0      | 0x25   | or       |
| 0x02   |        | j        |
| 0x04   |        | beq      |
| 0x05   |        | bne      |
| 0x08   |        | addi     |
| 0x0C   |        | andi     |
| 0x23   |        | lw       |
| 0x2B   |        | sw       |



|     |      |              |   |
|-----|------|--------------|---|
| L1: | lw   | \$6,0(\$7)   | 3 |
|     | and  | \$8,\$6,\$5  | 4 |
|     | beq  | \$8,\$0,L2   | 3 |
|     | sw   | \$8,\$4(\$7) | 4 |
|     | addi | \$7,\$7,8    | 4 |
|     | j    | L1           | 3 |
| L2: | ...  |              |   |

25. Para as 3 primeiras instruções do trecho de código apresentado na Figura 1, os sinais de controlo representados no seguinte diagrama temporal correspondem, pela ordem indicada, a:
- "ALUSelB", "ALUOp" e "PCSource".
  - "PCSource", "ALUOp" e "ALUSelB".
  - "PCSource", "ALUSelB" e "ALUOp".
  - "ALUSelB", "PCSource" e "ALUOp".



| ALUOp |        |
|-------|--------|
| 00    | ADD    |
| 01    | SUB    |
| 1X    | R-Type |

1111 1111 0000 0001 1000 0000 1111 1111

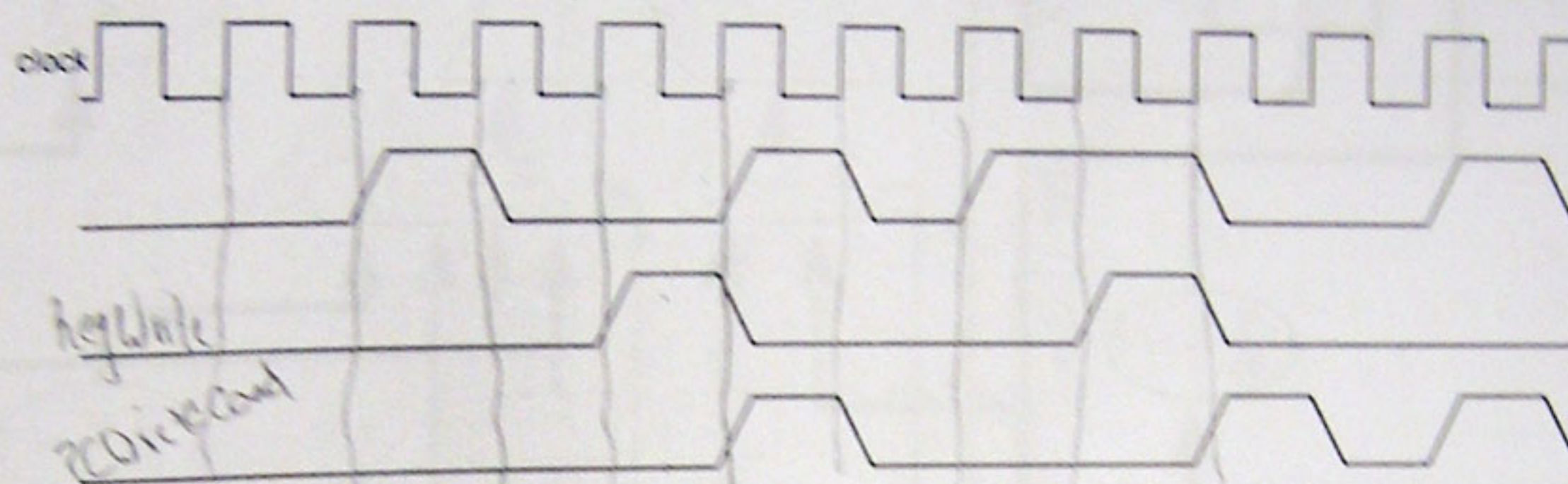
0001 0000 0000 0001 0000 0000 1010 0100

0000 0000 0000 0000 1000

\$6-

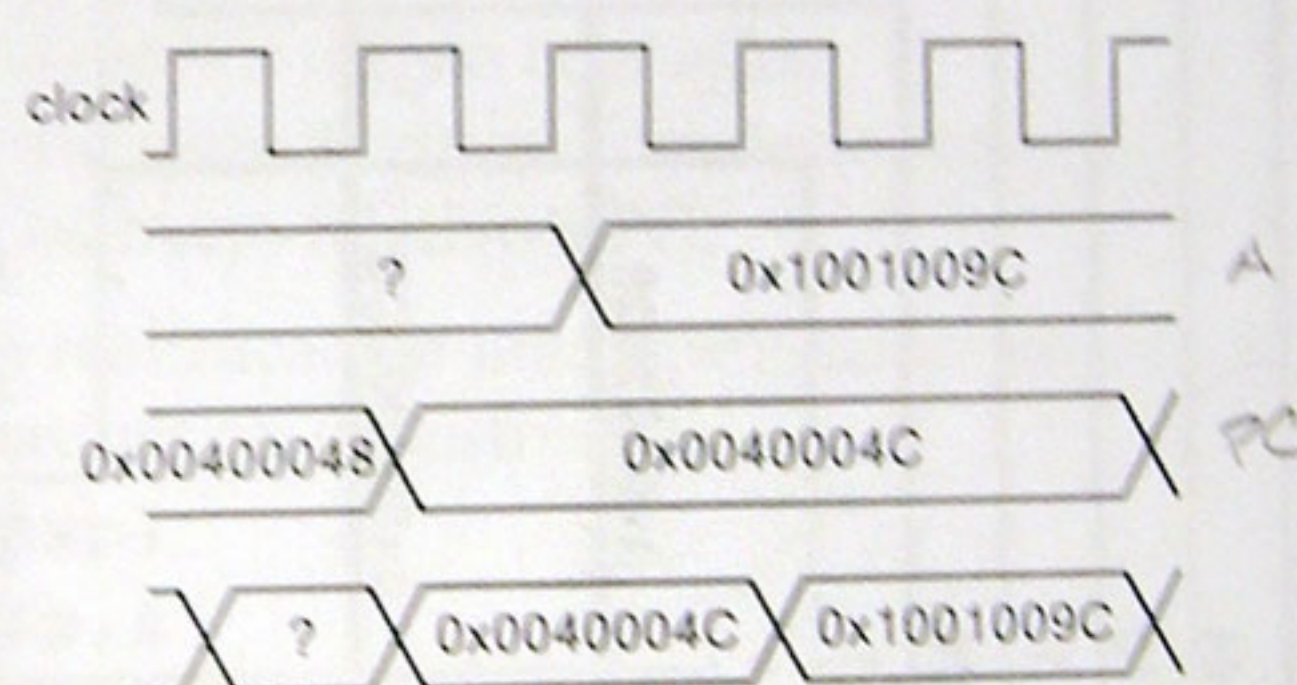


26. Também para as 3 primeiras instruções do trecho de código apresentado na Figura 1, os sinais de controlo representados no seguinte diagrama temporal correspondem, pela ordem indicada, a:
- "RegWrite", "PCWriteCond" e "RegDst".
  - "RegDst", "RegWrite" e "PCWriteCond".
  - "PCWriteCond", "RegWrite" e "RegDst".
  - "RegDst", "PCWriteCond" e "RegWrite".



27. Para a primeira instrução do trecho de código apresentado na Figura 1, e supondo que os valores dos registos do CPU são os que se indicam na mesma figura, os sinais do datapath representados no seguinte diagrama temporal correspondem, pela ordem indicada, a:

- "A", "InstRegister" e "PC".
- "B", "PC" e "ALUOut".
- "A", "PC" e "ALUOut".
- Nenhuma das anteriores.



28. Face aos valores presentes no segmento de dados (tabela da esquerda) e nos registos, o número total de ciclos de relógio que demora a execução completa do trecho de código apresentado, numa implementação *multi-cycle* do MIPS, é (desde o instante inicial do *instruction fetch* da primeira instrução até ao momento em que vai iniciar-se o *instruction fetch* da instrução presente em "L2:");

- 58 ciclos de relógio.
- 12 ciclos de relógio.
- 6 ciclos de relógio.
- 35 ciclos de relógio.

Zona de rascunho:

$$0x00400048 - 0x0040004C$$

$$? \quad ? \quad 100100AC$$

$$? \quad ? \quad 100100AC$$

$$? \quad 0x0040004C \quad 0x0040004C$$

$$\begin{array}{r} 11712 \\ 1 \quad 5812 \\ \hline 0 \quad 2912 \\ 1 \quad 1412 \\ \hline 0 \quad 712 \\ 1 \quad 312 \end{array}$$

$$\begin{array}{r} 1100 \\ 0101 \\ \hline 0100 \\ \hline 0004 \end{array}$$

$$-20L$$

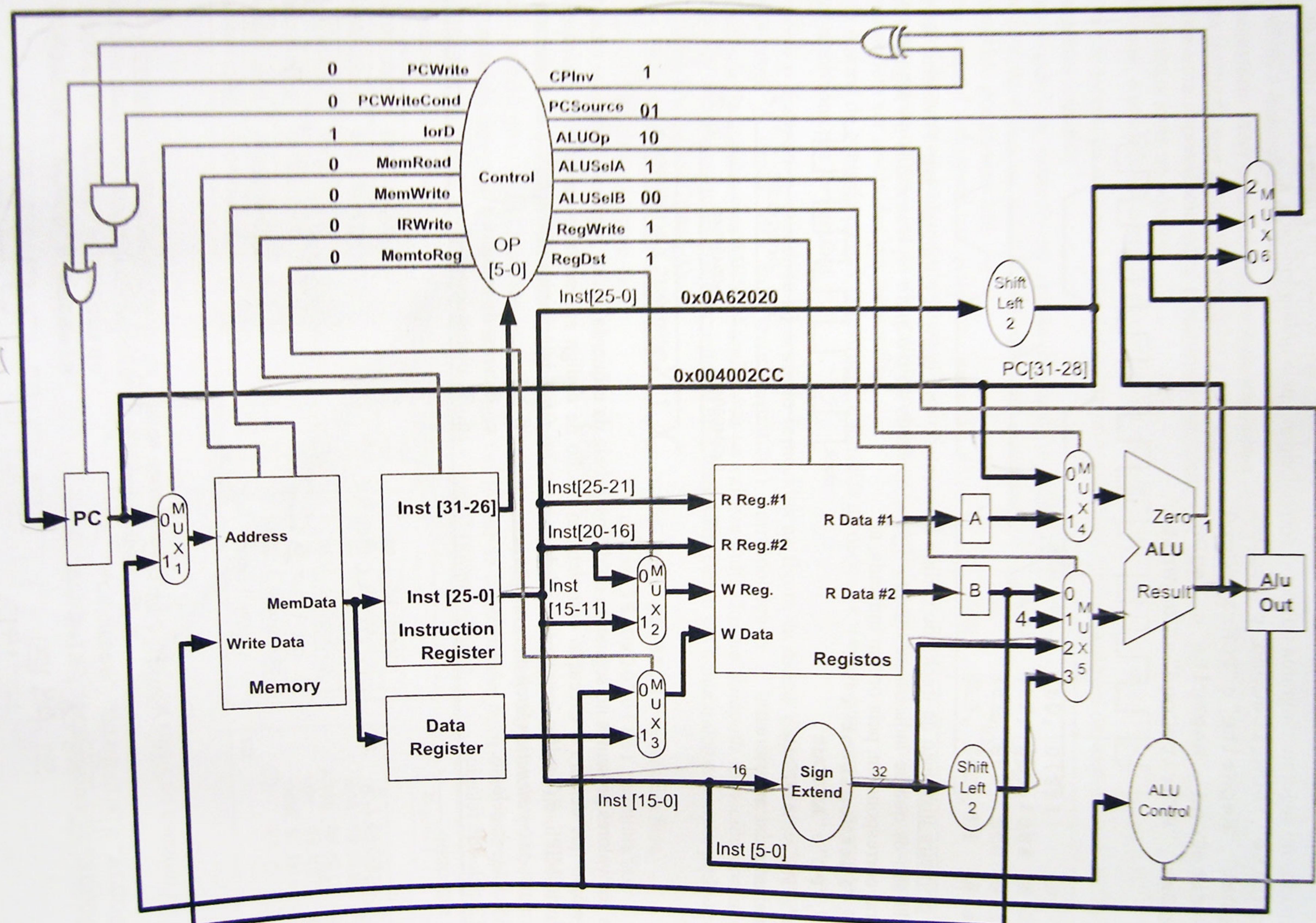
$$-4$$

$$\begin{array}{r} 0000 - 0 \\ 0001 - 1 \\ 0010 - 2 \\ 0011 - 3 \\ 0100 - 4 \\ 0101 - 5 \\ 0110 - 6 \\ 0111 - 7 \\ 1000 - 8 \\ 1001 - 9 \\ 1010 - 10 \\ 1011 - 11 \\ 1100 - 12 \end{array}$$



ALUSel13  
ALUOp  
PC Source

Figura 2





1. Uma arquitectura do tipo *Harvard* é caracterizada por:

### Teste 1

b. permitir o acesso a instruções e dados no mesmo ciclo de relógio

|    | a | b | c | d |
|----|---|---|---|---|
| 1  |   | X |   |   |
| 2  | X |   |   |   |
| 3  |   |   |   | X |
| 4  |   |   | X |   |
| 5  |   |   | X |   |
| 6  |   |   |   | X |
| 7  |   |   | X |   |
| 8  | X |   |   |   |
| 9  |   |   | X |   |
| 10 | X |   |   |   |
| 11 |   | X |   |   |
| 12 |   |   |   | X |
| 13 |   |   |   | X |
| 14 |   | X |   |   |
| 15 | X |   |   |   |
| 16 | X |   |   |   |
|    | a | b | c | d |
| 17 |   | X |   |   |
| 18 |   |   | X |   |
| 19 |   | X |   |   |
| 20 |   |   |   | X |
| 21 | X |   |   |   |
| 22 |   |   | X |   |
| 23 |   | X |   |   |
| 24 |   |   |   | X |
|    | a | b | c | d |
| 25 | X |   |   |   |
| 26 |   |   |   | X |
| 27 |   | X |   |   |
| 28 |   |   | X |   |

### Teste 2

d. permitir o acesso a instruções e dados no mesmo ciclo de relógio.

|    | a | b | c | d |
|----|---|---|---|---|
| 1  |   |   |   | X |
| 2  |   |   | X |   |
| 3  |   | X |   |   |
| 4  | X |   |   |   |
| 5  | X |   |   |   |
| 6  |   | X |   |   |
| 7  | X |   |   |   |
| 8  |   |   |   | X |
| 9  | X |   |   |   |
| 10 |   |   |   | X |
| 11 |   |   | X |   |
| 12 |   | X |   |   |
| 13 | X |   |   |   |
| 14 |   |   | X |   |
| 15 |   |   |   | X |
| 16 |   |   | X |   |
|    | a | b | c | d |
| 17 |   |   |   | X |
| 18 | X |   |   |   |
| 19 |   |   |   | X |
| 20 |   | X |   |   |
| 21 |   |   | X |   |
| 22 |   | X |   |   |
| 23 |   |   |   | X |
| 24 |   | X |   |   |
|    | a | b | c | d |
| 25 |   |   | X |   |
| 26 |   | X |   |   |
| 27 |   |   | X |   |
| 28 | X |   |   |   |

I