



Nome: _____

Notas importantes!

- Verifique, para todas as questões, qual a resposta correta e assinala com uma cruz a sua escolha na tabela ao lado. Por cada resposta incorreta será descontada à cotação global, no máximo, 1/3 da cotação da respetiva pergunta.
- Pode usar até um máximo de **4** respostas duplas (por cada dupla: 0 certas desconta até 2/3, 1 certa conta até 7/8). Se usar mais de **4** duplas, serão aceites as 4 primeiras e as restantes serão consideradas respostas erradas.
- Durante a realização do teste não é permitida a permanência junto do aluno, mesmo que desligado, de qualquer dispositivo eletrónico não expressamente autorizado (nesta lista incluem-se calculadoras, telemóveis e *smartwatches*). A sua deteção durante a realização do exame implica a imediata anulação do mesmo.

Grupo I

- Uma arquitetura do tipo Harvard é caracterizada por:
 - ter zonas distintas de endereçamento para dados e para código dentro da mesma memória.
 - permitir o acesso a instruções e dados no mesmo ciclo de relógio.
 - partilhar a mesma memória entre dados e instruções.
 - ter dois barramentos de dados e um barramento de endereços.
- O formato de instruções tipo “I” da arquitetura MIPS é usado nas instruções:
 - de deslocamento em que *imm* identifica o número de deslocamentos a efetuar.
 - aritméticas em que ambos os operandos estão armazenados em registos.
 - de salto incondicional.
 - de acesso à memória de dados externa.
- O resultado da instrução **mult \$t0, \$t1** é representável em 32 bits se:
 - HI** for uma extensão do bit mais significativo de **LO**.
 - HI = 0x00000000**.
 - HI** for diferente de zero.
 - HI = 0xFFFFFFFF**.
- Numa memória com uma organização do tipo *byte-addressable*:
 - a. cada posição de memória é identificada com um endereço com a dimensão de 1 byte.
 - b. o acesso apenas pode ser efetuado por instruções que transferem 1 byte de informação.
 - c. não é possível o armazenamento de palavras com dimensão superior a 1 byte.
 - d. a cada endereço está associado um registo com capacidade de armazenar 1 byte.
- Considere uma arquitetura em que o respetivo **ISA** especifica uma organização de memória do tipo *word-addressable* (*word* = 16 bits). Sabendo que o espaço de endereçamento do processador é de 30 bits, qual a dimensão máxima de memória que é possível acomodar nesta arquitetura, expressa em bits:

- 1 Gbit.**
 - 256 Mbit.**
 - 16 Gbit.**
 - 2 Gbit.**

$2^{30} \times 16$
- A arquitetura MIPS é do tipo “Load-Store”. Isso significa que:
 - nesta arquitetura foi dada especial importância à implementação das instruções *Load* e *Store*, de forma a não comprometer o desempenho global.
 - os operandos das operações aritméticas e lógicas apenas podem residir em registos internos.
 - os operandos das operações aritméticas e lógicas podem residir na memória externa.
 - as instruções de *Load* e *Store* apenas podem ser usadas imediatamente antes de operações aritméticas e lógicas.

	a	b	c	d
1		X		
2				X
3	X			
4	X			
5			X	
6		X		
7				X
8				X
9				
10	X			
11			X	
12		X		
13				X
14	X			
15	X			
16			X	
17			X	
18	X			
19				X
20		X		
21	X			
	a	b	c	d
22				X
23			X	
24	X			
25		X		
26			X	
27				
28				
29	X			
30				
31				
32				

7. A representação normalizada e arredondada para o par mais próximo de acordo com o formato IEEE754 precisão simples do número $100,110110001000010010110_2$ é:
- $1,000000000000000000000000 \times 2^{-2}$
 - $1,00110110100001000101 \times 2^2 \leftarrow$
 - $1,001101100000000000000110 \times 2^{-2}$
 - $1,001101100000000000000110 \times 2^2 \leftarrow$
8. Considerando que no endereço de memória acedido pela instrução **lb \$t0, 0x00FF(\$t1)** está armazenado o valor **0x82**, o valor armazenado em **\$t0** no final da execução dessa instrução é:
- 0x000000FF**
 - 0xFFFFFFF82**
 - 0x0000FF82**
 - 0x00000082**
9. Admita que o valor armazenado no registo **\$f0=0xFF800000** representa uma quantidade real em precisão simples. O valor equivalente em notação científica será:
- -1.0×2^{128}
 - NaN**
 - infinito**
 - $-0.000000000000000000000000 \times 2^{-127}$
10. Uma implementação *pipelined* de uma arquitetura possui, relativamente a uma implementação *single-cycle* da mesma, a vantagem de:
- diminuir o tempo necessário para realizar todas as operações de uma instrução.
 - aumentar o débito de execução das instruções de um programa.
 - tirar partido do facto de algumas instruções precisarem de menos ciclos de relógio do que outras.
 - permitir a redução do hardware necessário para a execução do mesmo *set* de instruções.
11. Numa implementação *single-cycle* da arquitetura MIPS:
- existem registos à saída dos elementos operativos fundamentais para guardar valores a utilizar no ciclo de relógio seguinte.
 - 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 memórias independentes para código e dados para possibilitar o acesso a ambos os tipos de informação num único ciclo de relógio.
 - todas as operações de leitura e escrita são síncronas com o sinal de relógio.
12. Nas instruções de acesso à memória da arquitetura MIPS é utilizado o modo de endereçamento:
- imediato.
 - tipo registo.
 - indireto por registo com deslocamento.**
 - indireto por registo.
13. O trecho de código que permite atribuir o valor **0xFF** à variável “i” indiretamente através do ponteiro “p” é:

a. <code>int i; int *p; p = &i; *p = 0xFF;</code>	b. <code>int i; int *p; p = *i; *p = 0xFF;</code>	c. <code>int i; int *p=0xFF; p = &i; i = *p;</code>	d. <code>int i; int *p; i = &p; *i = 0xFF;</code>
--	--	--	--

14. A instrução virtual **la \$t0, label** da arquitetura MIPS, em que **label** corresponde ao segundo endereço do segmento de dados do **MARS**, decompõem-se na seguinte sequência de instruções nativas:
- lui \$1, 0x1001** seguida de **ori \$t0, \$1, 0x0001**.
 - ori \$t0, \$0, 0x001** seguida de **lui \$1, 0x1001**.
 - lui \$1, 0x0040** seguida de **ori \$t0, \$1, 0x0001**.
 - ori \$t0, \$0, 0x0001** seguida de **lui \$1, 0x0040**.
15. A deteção de *overflow* numa operação de adição de números inteiros com sinal faz-se através:
- da avaliação do **bit** mais significativo do resultado.
 - do **xor** entre o **carry in** e o **carry out** da célula de 1 **bit** mais significativa do resultado.
 - do **xor** entre os 2 **bits** mais significativos do resultado.
 - da avaliação do **carry out** do **bit** mais significativo do resultado.
16. A unidade de controlo de uma implementação **multi-cycle** da arquitetura MIPS:
- 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 **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 um elemento combinatório que gera os sinais de controlo em função do campo **funct** do código máquina da instrução.
17. Considere uma implementação **multi-cycle** da arquitetura MIPS. Na segunda e terceira fases de execução de uma instrução de salto condicional ("**beq/bne**"), a ALU é usada, pela ordem indicada, para:
- 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**.
 - 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).
18. A frequência de relógio de uma implementação **pipelined** da arquitetura MIPS:
- é limitada pelo **maior** dos tempos de atraso dos elementos operativos Memória, ALU e *File Register*.
 - 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*.
 - é definida por forma a evitar *stalls* e *delay slots*.
19. A técnica de *forwarding/bypassing* num processador MIPS **pipelined** permite:
- escrever o resultado de uma instrução no *Register File* antes de esta chegar à etapa **WB**.
 - trocar a ordem de execução das instruções de forma a resolver *hazards* de dados.
 - utilizar, como operando de uma instrução, um resultado produzido por outra instrução que se encontra numa etapa mais recuada do pipeline.
 - utilizar, como operando de uma instrução, um resultado produzido por outra instrução que se encontra numa etapa mais avançada do pipeline.
20. Um *hazard* de controlo numa implementação **pipelined** de um processador ocorre quando:
- existe uma dependência entre o resultado calculado por uma instrução e o operando usado por outra que segue mais atrás do *pipeline*.
 - um dado recurso de hardware é necessário para realizar, no mesmo ciclo de relógio, duas ou mais operações relativas a instruções que se encontram 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.
 - a unidade de controlo desconhece o *opcode* da instrução que se encontra na etapa **ID**.

21. Numa implementação *single cycle* da arquitetura 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; preparação para escrita - 6ns;

File register: leitura - 3ns; preparação para escrita - 2ns;

Unidade de controlo: 2ns;

ALU (qualquer operação): 8ns;

Somadores: 4ns; Outros: 0ns

- a. 32,25 MHz (T=31ns). \leftarrow $l_w: T_{RM} + \max(RRF, CTML, SE) + ALU + RM + WRF$
 $9 + 3 + 8 + 9 + 2 = 31ns$
 b. 31,25 MHz (T=32ns).
 c. 29,41 MHz (T=34ns).
 d. 25,00 MHz (T=40ns).

Grupo II

22. A instrução virtual **bgt \$t8,\$t9,target** da arquitetura MIPS decompõe-se na seguinte sequência de instruções nativas:

- a. **slt \$1,\$t9,\$t8** seguida de **beq \$1,\$0,target**.
 b. **slt \$1,\$t8,\$t9** seguida de **beq \$1,\$0,target**.
 c. **slt \$1,\$t8,\$t9** seguida de **bne \$1,\$0,target**.
 d. **slt \$1,\$t9,\$t8** seguida de **bne \$1,\$0,target**. ✗

23. Admita que se pretende inicializar o conteúdo do registo **\$f4** com a quantidade real **2.0**₁₀ codificada em precisão simples. A sequência de instruções *Assembly* que efetua esta operação é:

a. li.s \$f0, 2.0 cvt.s.w \$f4, \$f0	b. li \$t0, 2 mtc1 \$t0, \$f0 mov.s \$f4, \$f0	c. li \$t2, 2 mtc1 \$t2, \$f4	d. lui \$t0, 0x4000 mtc1 \$t0, \$f4
--	--	---	---

$mtc1 \$ac, Dat \mid mfc1 Dat, \ac

24. Considerando que **\$f2=0x3A600000** e **\$f4=0xBA600000**, o resultado da instrução **sub.s \$f0,\$f2,\$f4** será:

- a. **\$f0=0x80000000**
 b. **\$f0=0x00000000**
 c. **\$f0=0x39E00000**
 d. **\$f0=0x3AE00000**

$$\begin{array}{r}
 0011 \ 1010 \ 0110 \\
 1011 \ 1010 \ 0110 \\
 0100 \ 0101 \ 1001 \\
 \hline
 0100 \ 0101 \ 1010
 \end{array}$$

0x80000000

25. 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:

- a. **6,25 x 2²**
 b. **-3,25 x 2¹** \leftarrow
 c. **-16,25 x 2¹**
 d. **-0,1625 x 2¹**

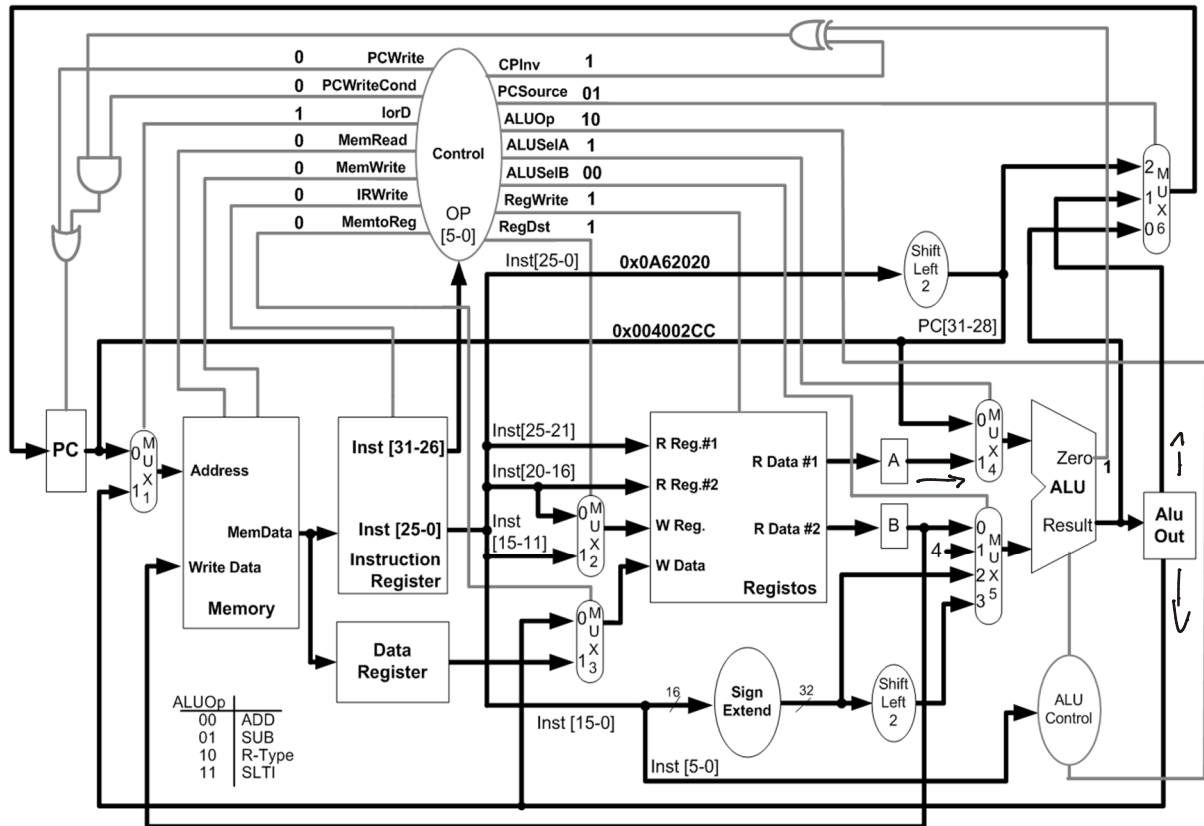
$1100 \ 0000 \ 1101 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000$

Neg

Exp: $1000 \ 0001 = 129 - 127 = 2$

$$\begin{aligned}
 -1,101 \times 2^2 &= -11,01 \times 2^1 \\
 &= -3,25 \times 2^1
 \end{aligned}$$

29. Considere o *datapath* e a unidade de controlo fornecidos na figura abaixo (com ligeiras alterações relativamente à versão das aulas teórico-práticas) correspondendo a uma implementação multi-cycle simplificada da arquitetura 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 aí presentes, pode-se concluir que está em execução uma das seguintes instruções:



- add \$4, \$5, \$6** na quarta fase.
- lw \$6, 0x2020 (\$5)** na terceira fase.
- add \$4, \$5, \$6** na terceira fase.
- lw \$6, 0x2020 (\$5)** na quinta fase.

30. Os processadores da arquitetura hipotética **HipCpu** implementam um total de 59 instruções. Todas as instruções são codificadas em 32 bits, num formato com 5 campos: *opcode*, três campos para identificar registos internos e um campo para codificar valores imediatos na gama [-4096, +4095]. Conclui-se, portanto:

- que o número de registos internos é 16 e o campo *opcode* tem 6 bits.
- que o número de registos internos é 8 e o campo *opcode* tem 8 bits.
- que o número de registos internos é 32 e o campo *opcode* tem 6 bits.
- que o número de registos internos é 16 e o campo *opcode* tem 7 bits.

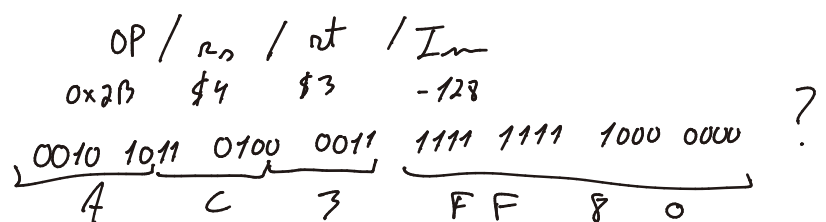


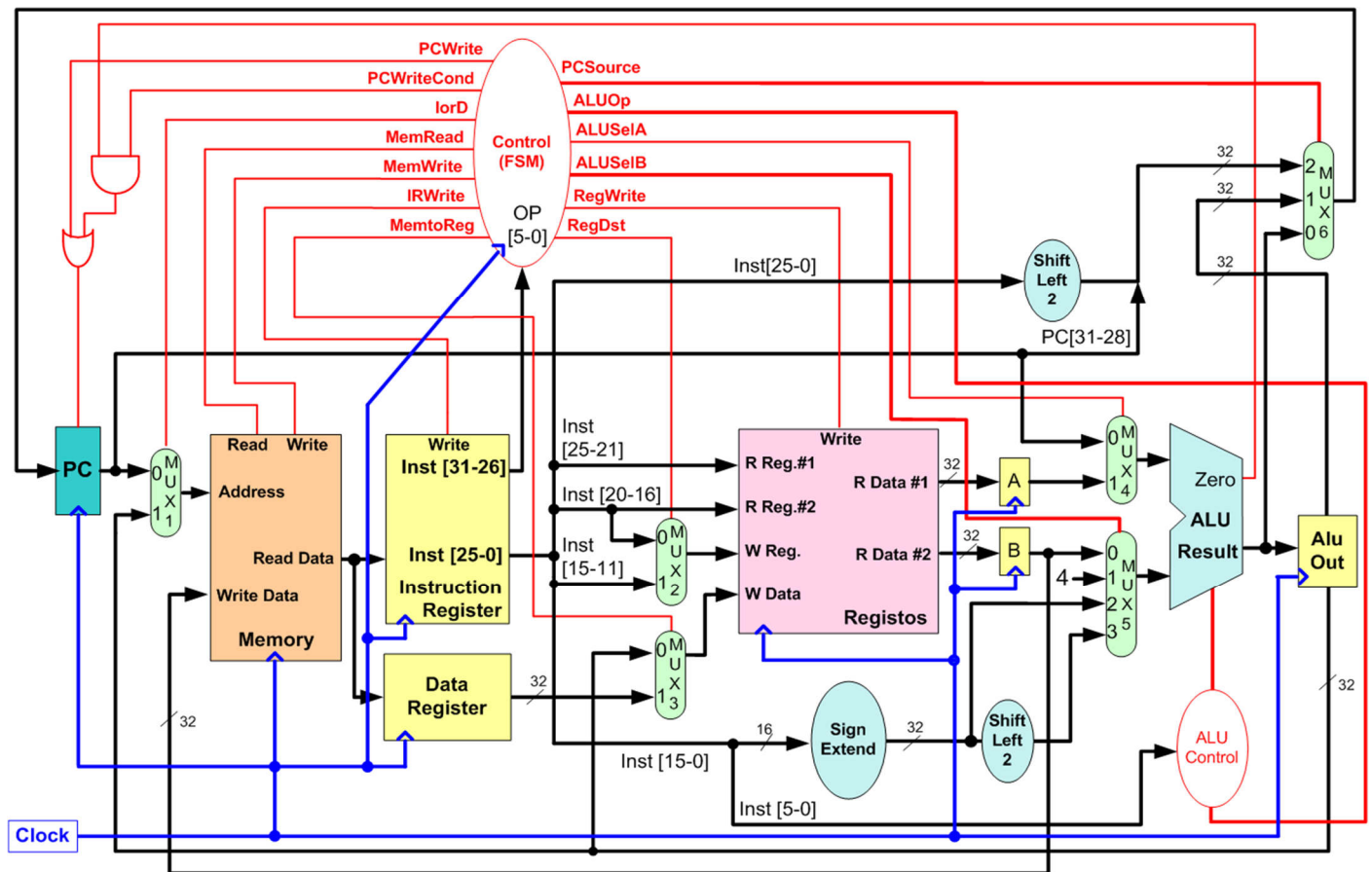
1 2 4 ~~8~~ 16 32 64 128 256 512 1024 ~~2048~~ 4096 8192 ..

$$32 - 14 = 18 \quad \begin{matrix} 13 \\ 2 \end{matrix} \quad ?$$

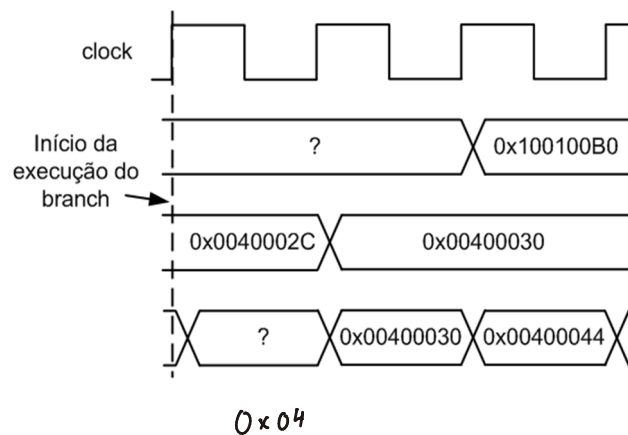
31. O código máquina da instrução **sw \$3, -128(\$4)**, representado em hexadecimal, é (considerando que, para esta instrução, *opcode* = **0x2B**):

- 0xAC64FF80**
- 0xAC83FF80**
- 0xAC838080**
- 0xAC648080**





32. Considere a instrução **beq \$5, \$6, L2** armazenada no endereço **0x0040002C**, a executar no *datapath multi-cycle* apresentado acima. Considere ainda o diagrama temporal abaixo apresentado. Sabendo que **\$5=0x1001009C** e **\$6=0x100100B0** e que o *opcode* da instrução é **0x04**, determine o código máquina, em hexadecimal, da instrução indicada.



- 0x10560005
- 0x40650014
- 0x10A60005
- 0x40560014