

Nome: \_\_\_\_\_



### Notas importantes!

1. 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.
2. 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.
3. 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

1. Uma arquitetura do tipo Harvard é caracterizada por: Harvard usa duas memórias independentes, ao contrário do modelo Von Neumann
  - a. ter zonas distintas de endereçamento para dados e para código dentro da mesma memória.
  - b. permitir o acesso a instruções e dados no mesmo ciclo de relógio.
  - c. partilhar a mesma memória entre dados e instruções.
  - d. ter dois barramentos de dados e um barramento de endereços.
2. O formato de instruções tipo "I" da arquitetura MIPS é usado nas instruções:
  - a. de deslocamento em que *imm* identifica o número de deslocamentos a efetuar.
  - b. aritméticas em que ambos os operandos estão armazenados em registos.
  - c. de salto incondicional.
  - d. de acesso à memória de dados externa. lw,sw,sb,lb, ....Tudo instruções tipo I
3. O resultado da instrução **mult \$t0, \$t1** é representável em 32 bits se:
  - a. **HI** for uma extensão do bit mais significativo de **LO**.
  - b. **HI** = **0x00000000**. Dependendo se é negativo ou positivo, O HI vai extender 1 ou 0.
  - c. **HI** for diferente de zero. O HI vai extender 1 ou 0.
  - d. **HI** = **0xFFFFFFFF**. Caso seja multu será sempre HI = 0x00000000
4. 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.
5. 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:
  - a. **1 Gbit.**
  - b. **256 Mbit.**
  - c. **16 Gbit.**
  - d. **2 Gbit.**
6. A arquitetura MIPS é do tipo "*Load-Store*". Isso significa que:
  - a. nesta arquitetura foi dada especial importância à implementação das instruções *Load* e *Store*, de forma a não comprometer o desempenho global.
  - b. os operandos das operações aritméticas e lógicas apenas podem residir em registos internos.
  - c. os operandos das operações aritméticas e lógicas podem residir na memória externa.
  - d. as instruções de *Load* e *Store* apenas podem ser usadas imediatamente antes de operações aritméticas e lógicas.

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

7. A representação normalizada e arredondada para o par mais próximo de acordo com o formato IEEE754 precisão simples do número  $100,110110000000000000010110_2$  é:
- a.  $1,000000000000000000000000 \times 2^{-2}$
- b.  $1,001101100000000000000101 \times 2^2$  IMPAR MAIS PROXIMO
- c.  $1,001101100000000000000110 \times 2^{-2}$
- d.  $1,001101100000000000000110 \times 2^2$  PAR MAIS PROXIMO
8. Considerando que no endereço de memória acessado pela instrução **lb \$t0, 0x00FF(\$t1)** está armazenado o valor **0x82**, o valor armazenado em **\$t0** no final da execução dessa instrução é:
- a. **0x000000FF**
- b. **0xFFFFFFF82**
- c. **0x0000FF82**
- d. **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:
- a.  $-1.0 \times 2^{128}$
- b. NaN
- c. **-infinito**
- d.  $-0.000000000000000000000000 \times 2^{-127}$
10. Uma implementação *pipelined* de uma arquitetura possui, relativamente a uma implementação *single-cycle* da mesma, a vantagem de:
- a. diminuir o tempo necessário para realizar todas as operações de uma instrução.
- b. **aumentar o débito de execução das instruções de um programa.**
- c. tirar partido do facto de algumas instruções precisarem de menos ciclos de relógio do que outras.
- d. permitir a redução do hardware necessário para a execução do mesmo *set* de instruções.
11. Numa implementação *single-cycle* da arquitetura MIPS:
- a. existem registos à saída dos elementos operativos fundamentais para guardar valores a utilizar no ciclo de relógio seguinte. Isto acontece apenas no pipelined
- b. 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. O single cycle usa também somadores
- c. **existem memórias independentes para código e dados para possibilitar o acesso a ambos os tipos de informação num único ciclo de relógio.**
- d. todas as operações de leitura e escrita são síncronas com o sinal de relógio. As leituras são assíncronas
12. Nas instruções de acesso à memória da arquitetura MIPS é utilizado o modo de endereçamento:
- a. imediato.
- b. tipo registo.
- c. **indireto por registo com deslocamento.**
- d. indireto por registo.
13. O trecho de código que permite atribuir o valor **0xFF** à variável “i” indiretamente através do ponteiro “p” é:

Atribui a p o endereço de memória de l e depois usa o ponteiro para guardar o valor

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

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:
- a. **lui \$1, 0x1001** seguida de **ori \$t0, \$1, 0x0001**. Primeiro lui e depois ori
  - b. **ori \$t0, \$0, 0x001** seguida de **lui \$1, 0x1001**. Primeiro endereço do segmento de dados  
0x10010000
  - c. **lui \$1, 0x0040** seguida de **ori \$t0, \$1, 0x0001**. Segundo - 0x10010001
  - d. **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:
- a. da avaliação do **bit** mais significativo do resultado.
  - b. do **xor** entre o **carry in** e o **carry out** da célula de 1 **bit** mais significativa do resultado.
  - c. do **xor** entre os **2 bits** mais significativos do resultado.
  - d. da avaliação do **carry out** do **bit** mais significativo do resultado. A unidade de controlo é uma máquina  
de estados com o número de estados  
dependente para cada instrução
16. A unidade de controlo de uma implementação **multi-cycle** da arquitetura MIPS:
- a. é uma máquina de estados com um número de estados igual ao número de fases da instrução mais longa.
  - b. é um elemento combinatório que gera os sinais de controlo em função do campo **opcode** do código máquina da instrução.
  - c. é uma máquina de estados em que o primeiro e o segundo estados são comuns à execução de todas as instruções.
  - d. 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:
- a. comparar os registos (operandos da instrução) e calcular o valor do **Branch Target Address**. Primeira fase  
PC+4
  - b. calcular o valor de **PC+4** e o valor do **Branch Target Address**. Segunda fase  
BTA
  - c. calcular o valor do **Branch Target Address** e comparar os registos (operandos da instrução). Terceira fase  
Comparar os registos
  - d. 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:
- a. é limitada pelo **maior** dos tempos de atraso dos elementos operativos Memória, ALU e **File Register**.
  - b. é limitada pelo **maior** dos atrasos cumulativos dos elementos operativos envolvidos na execução da instrução mais longa.
  - c. é limitada pelo **menor** dos tempos de atraso dos elementos operativos Memória, ALU e **File Register**.
  - d. é definida por forma a evitar *stalls* e *delay slots*.
19. A técnica de *forwarding/bypassing* num processador MIPS **pipelined** permite:
- a. escrever o resultado de uma instrução no **Register File** antes de esta chegar à etapa **WB**.
  - b. trocar a ordem de execução das instruções de forma a resolver *hazards* de dados.
  - c. utilizar, como operando de uma instrução, um resultado produzido por outra instrução que se encontra numa etapa mais recuada do pipeline.
  - d. 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:
- a. existe uma dependência entre o resultado calculado por uma instrução e o operando usado por outra que segue mais atrás do **pipeline**.
  - b. 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**.
  - c. 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.
  - d. 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

$$T_{rm} + T_{wf} + T_{rm} + T_{alu} + \max(T_{cntl}, T_{sse}, T_{rrf})$$

$$9 + 2 + 9 + 8 + 3 = 31 \text{ ns}$$

- a. 32,25 MHz (T=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**<sub>10</sub> codificada em precisão simples. A sequência de instruções *Assembly* que efetua esta operação é:

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

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

- a. **\$f0=0x80000000** 0x3A600000 - 0011 1010 0110 0000 0000 0000 0000 1.1100  
- 0 01110100 1100 .... - (- 1.1100)  
01110100 corresponde a 116 11.100  
b. **\$f0=0x00000000**  
c. **\$f0=0x39E00000** 116 - 127 = -11  
1.1100 \* 2<sup>-11</sup> 11.100 \* 2<sup>-11</sup>  
d. **\$f0=0x3AE00000** 0xBA600000 - 1011 1010 0110 0000 0000 0000 0000 1.1100 \* 2<sup>-10</sup>  
-1.1100 \* 2<sup>-11</sup> 0011 1010 1110 0000  
0x3AE00000

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 2<sup>2</sup>**  
b. **-3,25 x 2<sup>1</sup>** 1 1000 0001 1010000  
129 - 127 = 2  
c. **-16,25 x 2<sup>1</sup>** - 1.101 \* 2<sup>2</sup>  
- 11.01 \* 2<sup>1</sup>  
d. **-0,1625 x 2<sup>1</sup>** -3.25

26. Considere 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 100MHz relativamente a uma *single-cycle* a operar a 20 MHz é de:

$$100 / 20 * 1/4 = 1.25$$

a. 1

b. 1,25

c. 5

d. 0,8

lw -> 0.2 \* 5 = 1  
sw -> 0.1 \* 4 = 0.4  
type-r -> 0.5 \* 4 = 2.0  
beq -> 0.15 \* 3 = 0.45  
j -> 0.05 \* 3 = 0.15

Soma = 4.0

Tome como referência as tabelas a seguir apresentadas. Admita que o valor presente no registo **\$PC** é **0x00400128** e corresponde ao endereço da primeira instrução do programa “**Prog. 1**”. Considere ainda a implementação *pipelined* da arquitetura MIPS que estudou nas aulas, com *delayed-branch* e *forwarding* para **EX (MEM/WB > EX)** e **EX/MEM > EX** e para **ID (EX/MEM > ID)**.

Endereço	Dados	Prog. 1
...	...	
0x4CCC	0x093B863D	L0: xor \$10,\$0,\$0
0x4CC8	0x14A0C373	xori \$6,\$0,0x4CC8
0x4CC4	0x26B51E8C	L1: lw \$2,0(\$6)
0x4CC0	0xD94AE173	lw \$3,4(\$6)
0x4CBC	0xC31748FE	xor \$4,\$2,\$3
0x4CB8	0x601F3212	nor \$4,\$4,\$0
0x4CB4	0x0B506C98	beq \$4,\$0,L2
0x4CB0	0x03C12972	add \$10,\$10,\$2
		j L1
		addi \$6,\$6,-4
		L2: sw \$10,-48(\$6)

Ciclo vai percorrer 3 vezes  
4 ciclos ate o 1 correr  
2 ciclos para entrar no loop  
20 ciclos para percorrer no loop  
6 vezes  
1 vez  
4 + 2 + 20 + 8 + 1  
35 ciclos de relógio

27. A execução completa do trecho de código fornecido (em **Prog. 1**), desde o *instruction fetch* da instrução referenciada pelo *label L0* até à conclusão da instrução referenciada pelo *label L2*, demora:

a. 35 ciclos de relógio.

b. 13 ciclos de relógio.

c. 26 ciclos de relógio.

d. 40 ciclos de relógio.

28. Admita que no instante zero, correspondente a uma transição ativa do sinal de relógio, vai iniciar-se o *instruction fetch* da primeira instrução. O valor à saída da ALU na conclusão do sexto ciclo de relógio, contado a partir do instante zero, é:

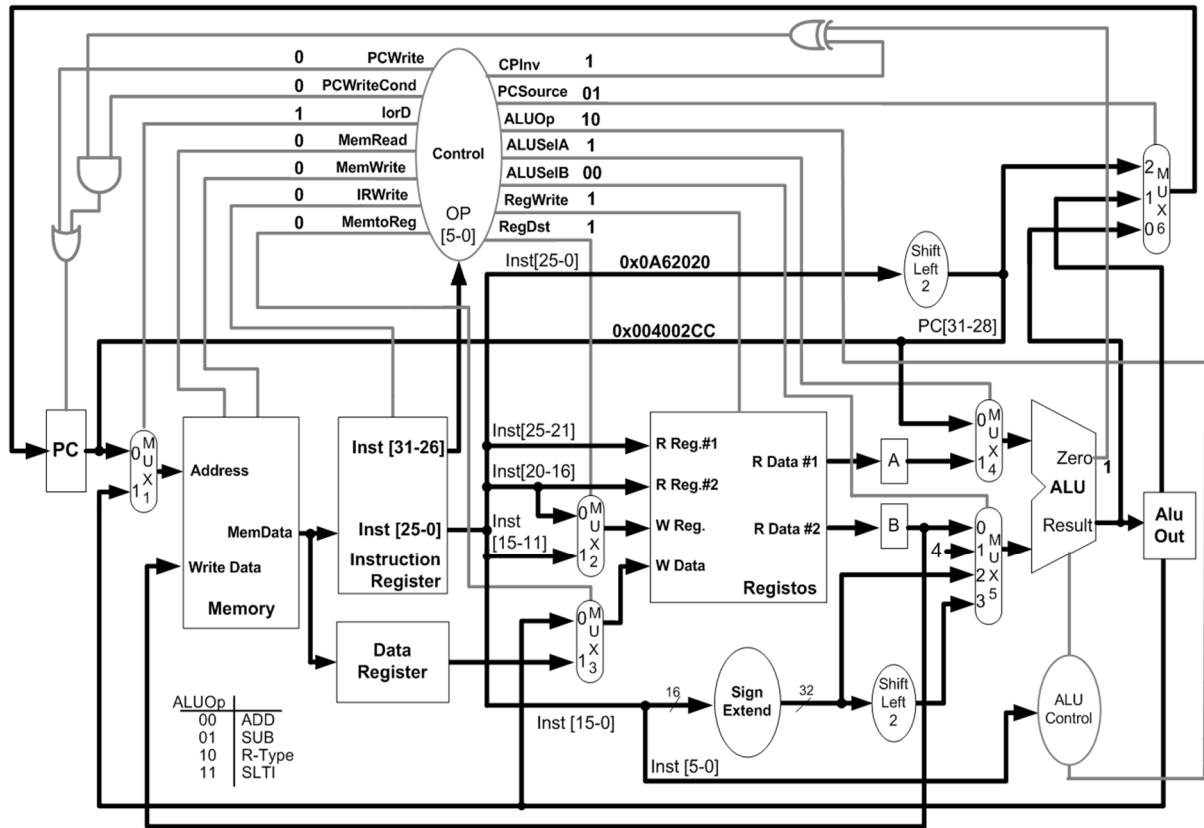
a. 0x00004CCC

b. 0x093B863D

c. 0x1D9B454E

d. 0x00004CC8

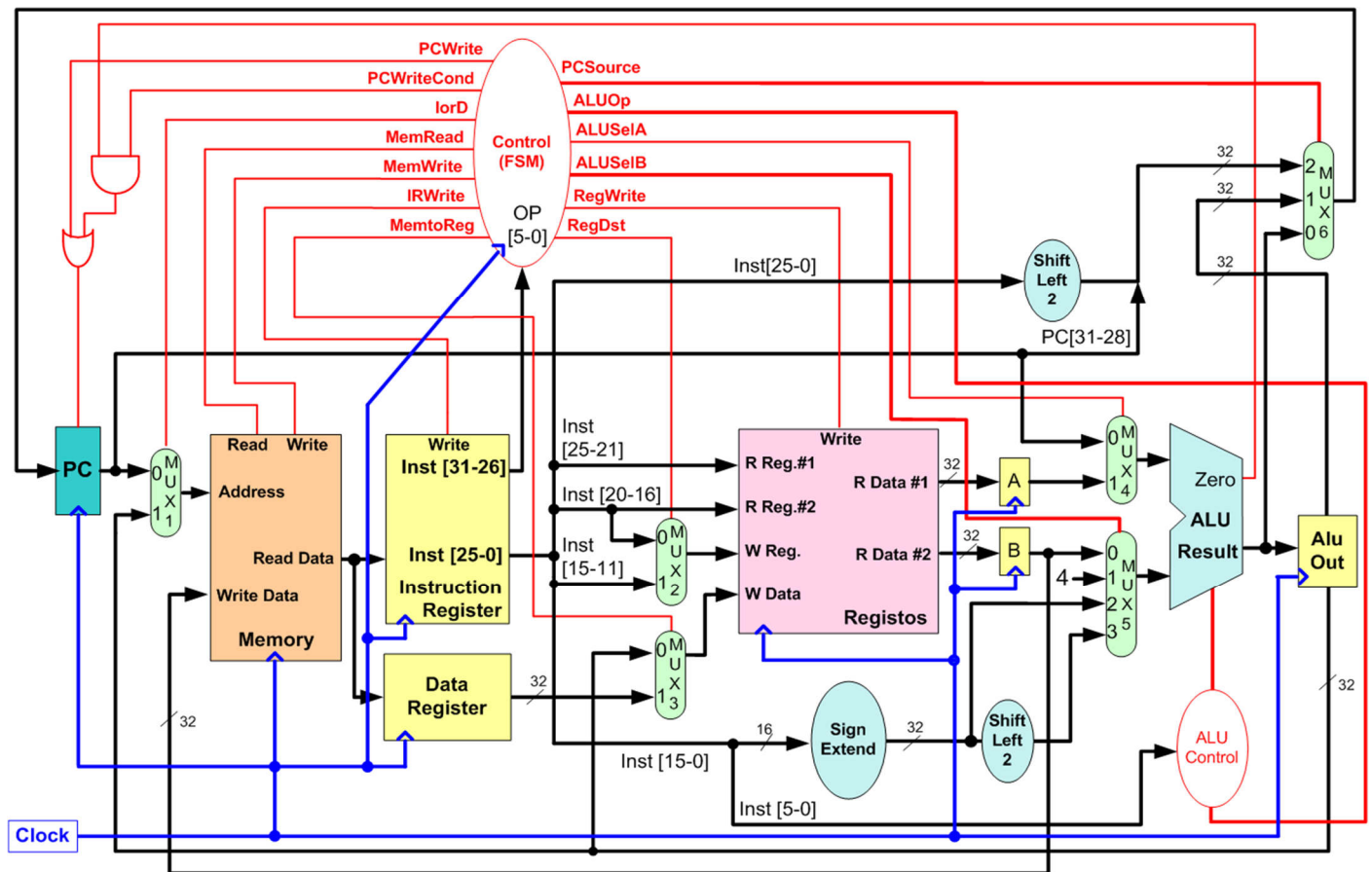
29. Considere o *datapath* e a unidade de controlo fornecidos na figura abaixo (com ligeiras alterações relativamente à versão das aulas teórico-práticas) correspondendo a uma implementação *multi-cycle* simplificada da arquitetura MIPS. Admita que os valores indicados no *datapath* fornecido correspondem 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:



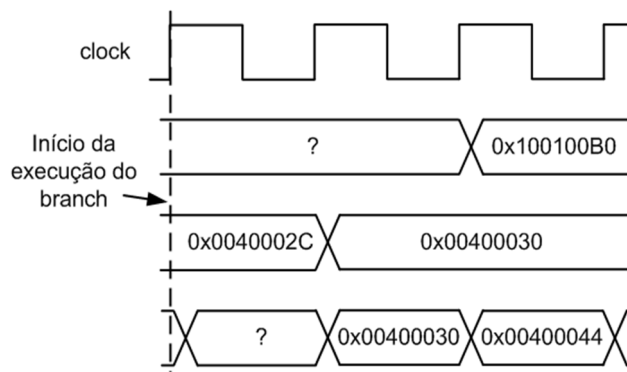
- a. **add \$4, \$5, \$6** na quarta fase.
  - b. **lw \$6, 0x2020 (\$5)** na terceira fase.
  - c. **add \$4, \$5, \$6** na terceira fase.
  - d. **lw \$6, 0x2020 (\$5)** na quinta fase.
30. Os processadores da arquitetura hipotética **HipCpu** implementam um total de 59 instruções. Todas as instruções 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:
- a. que o número de registos internos é 16 e o campo *opcode* tem 6 bits.
  - b. que o número de registos internos é 8 e o campo *opcode* tem 8 bits.
  - c. que o número de registos internos é 32 e o campo *opcode* tem 6 bits.
  - d. **que o número de registos internos é 16 e o campo *opcode* tem 7 bits.**
31. O código máquina da instrução **sw \$3, -128 (\$4)**, representado em hexadecimal, é (considerando que, para esta instrução, *opcode* = **0x2B**):
- a. **0xAC64FF80**
  - b. **0xAC83FF80**
  - c. **0xAC838080**
  - d. **0xAC648080**

1010 1100 1000 0011 1111 1111 1000





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.



- a. **0x10560005**
- b. **0x40650014**
- c. **0x10A60005**
- d. **0x40560014**

0001 0000 1010 0110