1. O principal modo de endereçamento utilizado na instrução beq $to, $t1, label é:

a. relativo ao PC.
b. pseudo-direto.
c. indireto por registo com deslocamento.
d. indireto por registo.

2. Considere a instrução jalr $t9, armazenada no endereço de memória 0x00400 63C. Admita que o conteúdo do registo $t9 = 0x00400680. Na conclusão da execução da instrução, o registo $ra:

a. terá armazenado o valor 0x00400684.
b. terá armazenado o valor 0x0040063C.
c. terá armazenado o valor 0x00400640.
d. terá armazenado o valor 0x00400680.

3. Na arquitetura básica de um sistema computacional o Data Bus, permite, por exemplo:

a. especificar a natureza de uma operação efetuada sobre a memória.
b. identificar, na memória, os dados a serem transferidos.
c. transferir dados entre os registos do CPU.
d. transferir o código máquina das instruções para o CPU.

4. Considere o seguinte segmento de código assembly:

```
li      $4, 0x011284C3
sw      $4, 0x08 ($2)
lb      $5, 0x0A ($2)
```

No final da execução do código anterior, o valor de $5, num MIPS little endian, é:

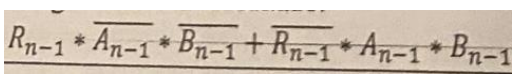
a. OxFFFFFF84
b. 0x00000084
c. 0x00000012
d. 0xFFFFFF01

5. Suponha que $3=0xFA11AAE8 e $4=0x00EEFF00. A instrução "nor $2, $3, $4" produz o seguinte resultado armazenado em $2:

a. OxFAFFAA17
b. 0x05000017
c. 0x0500FFE8
d. OXFAFFFFE8

6. A deteção de overflow numa operação de adição de dois números inteiros [R=A+B], com R, A e B codificados em complemento para dois, pode fazer-se através:

a. do xor entre os 2 bits mais significativos do resultado.
b. da avaliação do carry out do bit mais significativo do resultado.
c. da avaliação do bit mais significativo do resultado.
d. da expressão booleana



7. Admita que a instrução and $to, $t0,4($t1) corresponderia a uma instrução nativa do instruction set da arquitetura MIPS, codificada com o tipo I. Nesse caso poderiamos concluir que:

a. estaríamos perante uma arquitetura do tipo Register Memory
b. as operações realizadas na ALU operam apenas sobre conteudos de registos intemos.
c. por usar um modelo Harvard, esta instrução poderia ser executada no pipeline de 5 estágios estudado nas aulas.
d. esta arquitetura permite, para a mesma instrução e num mesmo ciclo de relógio, operar sobre a ALU e efetuar uma operação de leitura da memória.

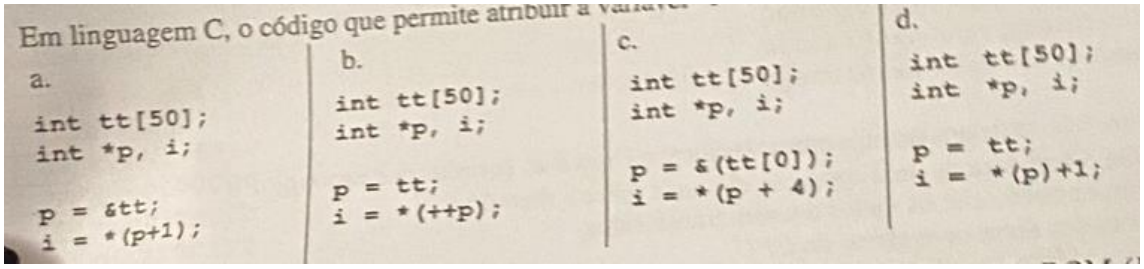
8.Considere que no endereço de memória 0x00400020 se encontra armazenada a instrução bgtz $2, label. Se o valor dos 16 bits menos significativos do código máquina dessa instrução for 0xFFFC, o endereço de label será:

a. 0x00400010
b. 0x00400020
c. 0x0040001C
d. 0x00400014

9. Considere que o barramento de endereços de um dado CPU é composto por 24 bits e a memória é do tipo word addressable (com word igual a 32 bits). Esta informação permite-nos conhecer:

a. quais os dados armazenados em cada posição de memória.
b. o espaço de endereçamento do CPU.
c. a dimensão da memória, expressa em bytes, instalada no sistema computacional.
d. qual a dimensão do barramento de dados do sistema computacional.

10. Em linguagem C, o código que permite atribuir à variável "i" o valor do elemento de indice 1 do array "tt":



RESPOSTA: B

11. Considere um sistema computacional em que existe uma memória do tipo FLASH_ROM (Read Only Memory), que armazena programas, e uma memória do tipo RAM que armazena dados. Sabendo que as duas memórias partilham o mesmo barramento de controlo, pode afirmar-se que estamos perante uma arquitetura:

a. do tipo Harvard com armazenamento não volátil de programas.
b. do tipo Harvard.
c. do tipo von Neumann.
d. mista Harvard/von Neumann.

12. A instrução jalr $t0 do set de instruções do MIPS permite:

a. evocar uma sub-rotina que se encontre em qualquer endereço múltiplo de 4 da memória, desde que o valor em $t0 tenha os 4 bits mais significativos iguais aos do atual PC.
b. evocar uma sub-rotina, armazenando o endereço de retorno no registo $t0 em vez de $ra.
c. evocar uma sub-rotina que comece em qualquer endereço múltiplo de 4 do espaço de endereçamento.
d. evocar uma sub-rotina sem necessidade de conhecer o endereço do label onde esta começa.

13. Na implementação multi-cycle de um processador MIPS, durante a execução da instrução sw $1,-4 ($2), a unidade de controlo ativa o sinal:

a. MemWrite no quinto ciclo de relógio.
b. MemRead no quarto ciclo de relógio.
c. IRWrite no primeiro ciclo de relógio.
d. RegWrite no quarto ciclo de relógio.

14. Considere o resultado da instrução multu $to, $t1. Este resultado só não é representável em 32 bits se:

a. HI for diferente de zero.
b. HI for igual a LO.
c. HI for uma extensão com sinal de LO.
d. HI=0x00000000.

15. Numa implementação single-cycle da arquitetura MIPS, semelhante à apresentada nas aulas, na transição ativa do sinal de relógio:

a. é atualizado o valor à saída do PC com o endereço da próxima instrução a executar,
b. é realizada a leitura síncrona da memória de instruções.
c. é lido o valor dos operandos a partir do banco de registos.
d. é realizada a leitura assíncrona do banco de registos.

16. Considere que o segmento de dados de um programa contém as seguintes diretivas:

```
L1:  .word 0, 0, -2024
L2:  .asciiz "RecursoJan24"
     .align 3
L3:  .space 8
L4:
```

Se o endereço correspondente ao label L1 for 0x100100c8, o endereço a que corresponde o label L4 é:

a. 0x10010100
b. 0x100100F0
c. 0x100100EC
d. 0x100100E9

17. Quando, numa arquitetura pipelined do MIPS é necessário fazer o instruction ferch de uma nova instrução e existe numa etapa mais avançada do pipeline uma instrução não concluída e que vai alterar o fluxo de execução, podemos afirmar que:

a. estamos perante um hazard de controlo.
b. estamos perante um hazard de dados.
c. estamos perante um hazard estrutural.
d. é necessário inserir dois ou mais stalls que permitam concluir a instrução que vai alterar o fluxo de execução.

18. A unidade de stalling de uma implementação pipelined da arquitetura MIPS é um bloco:

a. combinatório, que gera os sinais de controlo que permitem congelar as instruções nos estágios IF e ID e inserir um NOP no estágio EX.
b. baseado numa máquina de estados que deteta situações em que é necessário inserir uma bubble no pipeline.
c. combinatório responsável pelo avanço/paragem das instruções em todos os estágios do pipeline.
d. combinatório responsável pela geração dos sinais de controlo que permitem o encaminhamento de um resultado para estágios mais avançados do pipeline.

19. Os processadores da arquitetura hipotética ZXIoT implementam um total de 63 instruções. Todas as instruções são codificadas com 32 bits, e um dos formatos tem 5 campos: opcode, três campos para identificar registos internos e um campo para codificar constantes no intervalo [-8192, +8191]. Considerando que o número de bits do campo opcode é apenas o necessário para codificar as 63 instruções, conclui-se que o número de registos é:

a. 8
b. 4
c. 32
d. 16

20. No formato de vírgula flutuante IEEE 754, precisão simples, um expoente negativo é codificado:

a. com um valor superior a 0 e inferior a 127.
b. com um valor que tem o bit mais significativo a 1.
c. com um valor igual ou superior a 128 e igual ou inferior a 255.
d. com um valor igual ou superior a 1023 e igual ou inferior a 2046.

21. No terceiro ciclo de relógio de execução da instrução beq, numa arquitetura MIPS multi-cycle, é efetuada na ALU a seguinte operação:

a. a concatenação dos conteúdos dos registos codificados respetivamente no campo RS e no campo RT.
b. a comparação entre o conteúdo do registo codificado no campo RS e o valor imm estendido para 32 bits.
c. a subtração correspondente à expressão [RS]-[RT].
d. a soma do program counter com o valor multiplicado por 4 do offset sign extended para 32 bits.

22. Admita que os valores armazenados nos elementos de estado do datapath da Figura 1, na terceira fase de execução da instrução lw $2,-8 ($5) são: PC=0x0040005C, $5=0x10010C8, $2=0x00003C24. Neste pressuposto, o valor disponível à saída do registo ALUOut, durante a quarta fase de execução, dessa instrução será:

a. 0x0040003C
b. 0x100100C0
c. 0x00400024
d. 0x10013C24

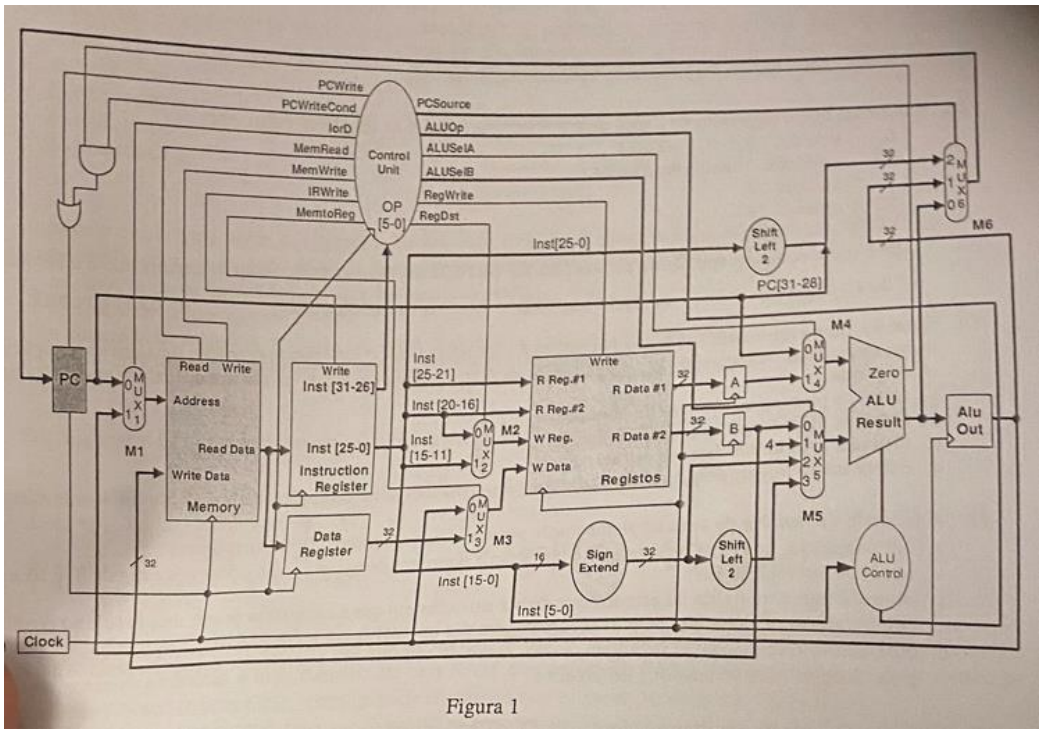


Figura 1

23. Suponha que, no datapath da Figura 1, imediatamente antes da ocorrência da transição ativa do relógio, os sinais de controlo PCWrite, ALUOP, RegDst, PCSource e RegWrite têm os valores '1', "10", '0', "10" e '0' respetivamente. Pode então concluir-se que:

a. está em execução uma instrução tipo R na terceira fase.
b. está em execução uma instrução na primeira fase.
c. está em execução uma instrução de salto condicional na terceira fase.
d. está em execução uma instrução de salto incondicional na terceira fase.

24. Se se pretendesse incluir suporte para a execução da instrução jal no datapath da Figura 1, uma das alterações a efetuar seria:

a. acrescentar uma entrada no multiplexer M4, ligada à entrada 2 do multiplexer M6.
b. acrescentar uma entrada no multiplexer M6, ligada à saída do registo A.
c. acrescentar uma entrada no multiplexer M2, ligada à constante 0x04.
d. acrescentar uma entrada no multiplexer M3, ligada à saída do registo PC.

25. A representação da quantidade +962, $5_{10} \times 10^{-2}$ no formato IEEE 754 precisão simples, é:

a. 0x40980000
b. 0x411A0000
c. 0xC1100000
d. 0x40F80000

26. A instrução virtual do MIPS que é implementada pela seguinte sequência de instruções nativas é:

```
ori $1, $0, 0x93
sltu $1, $1, $5
bne $1, $0, L1
```

a. bgtu $5,0x93, L1
b. bltu $5,0x93, L1
c. bleu $5,0x93, L1
d. bgeu $5,0x93, L1

27. Numa implementação pipelined da arquitetura MIPS, o hazard de dados existente na sequência de instruções

```
lw $1,0 ($2)
add $2,$3,$5
beq $1, $3, target
```

pode ser resolvido:

a. sem recurso a stalling, com forwarding de EX/MEM para EX.
b. com stall durante 1 ciclo de relógio.
c. com stall durante 1 ciclo de relógio seguido de forwarding de MEM/WB para ID,
d. com stall durante 1 ciclo de relógio seguido de forwarding de EX/MEM para ID.

28. O seguinte trecho de código, a executar sobre uma implementação pipelined da arquitetura MIPS com delayed branches, apresenta os seguintes hazards:

```
label:  lw $to, 0 ($t1)
        add $t2, $t1, $t4
        addi $t1, $t1, 4
        beq $to, $t2, label
        nop
```

a. um hazard de controlo na quarta instrução e um hazard de dados na segunda instrução que não pode ser resolvido por forwarding.
b. um hazard de controlo na quarta instrução e hazards de dados na segunda e quarta instruções que podem ser resolvidos por forwarding.
c. um hazard estrutural na primeira instrução e um hazard de controlo na quarta instrução.
d. um hazard de controlo na quarta instrução e um hazard de dados na quarta instrução que pode ser resolvido por forwarding.

29. O código máquina da instrução lbu $18,-16 ($3), representado em hexadecimal, é (considerando que, para esta instrução, opcode = 0x24):

a. 0x9243FFF8
b. 0x9083FFF0
c. 0x9072FFF0
d. 0x92E3FFF8

30. Tome como referência as tabelas a seguir apresentadas. Admita que o valor presente no registo PC é 0x0040005C e corresponde ao endereço da primeira instrução do programa. Considere ainda a implementação pipelined da arquitetura MIPS que estudou nas aulas, com delayed-branch e forwarding para EX (MEM/WB→EX, EX/MEM→EX) e para ID (EX/MEM→ID).

Endereço	Dados
0x684C	0x0938863D
0x6848	0x14A0C373
0x6844	0xC31748FE
0x6840	0xFFFFFF5A
0x683C	0x26B51E8A
0x6838	0x0F193ABA
0x6834	0x0B506C98
0x6830	0x03C1297E

```
L0:  addi  $4,$0,0x0A5
     xor   $2,$2,$2
     addi  $3,$0,0x6B48
L1:  lw    $5,0($3)
     add   $2,$2,$5
     nor   $6,$5,$4
     beq   $6,$0,L2
     nop
     j     L1
     addiu $3,$3,-4
L2:  sw    $2,-64($3)
```

A execução completa do trecho de código fornecido, desde o instruction fetch da instrução referenciada pelo label L0 até à conclusão da instrução referenciada pelo label L2, demora:

a. 45 ciclos de relógio.
b. 38 ciclos de relógio.
c. 33 ciclos de relógio.
d. 42 ciclos de relógio.

31. Na situação apresentada na questão anterior, 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 oitavo ciclo de relógio, contado a partir do instante zero, é:

a. 0x00006848
b. 0x00000000
c. 0x093B663D
d. Ox14A0C373

32. Considerando que $f4=0xBE300000 c $f6=0x40800000, o resultado armazenado em $f8 pela instrução mul.s $f8, $f4, $f6 é:

a. $f8 = 0xBF200000
b. $f8 = 0xBF300000
c. $f8 = 0x40100000
d. $f8 = 0x3F480000

33. A quantidade de memória em bytes, reservada pelo conjunto das directivas da figura do lado correspondendo a (La-L1), é:

a. 13
b. 27
c. 20
d. 18

34. 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.

35. Um endereço de memória externa num sistema computacional é:

a. a gama de posições de memória que o CPU pode referenciar.
b. um número único que identifica cada posição de memória.
c. a informação armazenada em cada posição.
d. um índice de um registo de uso geral.

36. Espaço de endereçamento de memória num sistema computacional é:

a. Um número único que identifica cada posição de memória.
b. A gama total de posições de memória que o CPU pode referenciar.
c. A informação armazenada em cada posição de memória.
d. A gama de posições de memória efectivamente disponíveis no sistema

37. Numa memória com uma organização do tipo byte-addressable:

a. A cada endereço está associado um dispositivo de armazenamento de 1 byte. Ou Cada registo de memória permite o armazenamento de 1 byte de informação.
b. Cada posição de memória é identificada com um endereço com a dimensão de 1 byte.
c. O acesso apenas pode ser efectuado por instruções que transferem 1 byte de informação.
d. Não é possível o armazenamento de quantidades com dimensao superior a 1 byte.

38. A arquitectura MIPS é caracterizada por:

a. possuir 16 registos de uso geral de 32 bits cada.
b. possuir um cpu capaz de realizar directamente operações aritméticas cujos operadores residem na memória externa.
c. ser do tipo load-store.
d. ter instruções de tamanho variável.

39. A arquitectura MIPS é caracterizada por:

a. possuir 32 registos de uso geral de 32 bits cada.
b. ser do tipo load-store.
c. possuir poucos formatos de instrução.
d. todas as anteriores.

40. A arquitectura MIPS é do tipo “Load-Store”. Isso significa que:

a. Os operandos das operações aritméticas e lógicas podem residir na memoria externa
b. Os operandos das operações aritméticas e lógicas apenas podem residir em registos internos
c. As instruções de Load e Store apenas podem ser usadas imediatamente antes de operações aritméticas e logicas.
d. Neste arquitectura foi dada especial importância à implementação das instruções Load e Store, de forma a não comprometer o desempenho global.

41. Na arquitectura MIPS, os campos de uma instrução do tipo “R” designam-se por:

a. “opcode”,”rs”,”rt” e “imm”.
b. “opcode” e “address”.
c. “opcode”,”rs”,”rt”,”rd”,”shamt” e “imm”.
d. nenhuma das anteriores.

42. Na arquitectura MIPS os campos de uma instrução tipo “I” designam-se por:

a. opcode, rs, rt e offset/imm
b. opcode, rs, rt, shamt e funct
c. opcode, rs, rt, rd e offset/imm
d. opcode, rs, rt, rd, shamt e funct

43. Nas instruções de acesso à memória da arquitectura MIPS é utilizado o modo de endereçamento:

a. indirecto por registo.
b. registo.
c. imediato.
d. Directo.

44. No MIPS, a instrução de salto incondicional indirecto através de registo:

a. É codificada usando o formato de codificação R.
b. É codificada usando o formato de codificação L.
c. É codificada usando o formato de codificação J.
d. A instrução em causa não existe.

45. 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

46. O formato de instruções tipo "I" da arquitectura MIPS é usado nas instruções de:

a. salto condicional.
b. aritméticas em que somente um dos operandos está armazenado num registo.
c. acesso à memória de dados externa.
d. todas as anteriores.

47. Na instrução lb da arquitectura MIPS, o operando é obtido através de endereçamento:

a. relativo ao PC com deslocamento.
b. imediato
c. directo
d. indirecto a registo com deslocamento.

48. Nas instruções tipo R da arquitectura MIPS é utilizado o modo de endereçamento:

a. indirecto por registo
b. Registo.

49. O modo de endereçamento utilizado na instrução sb $8,-8($s8) é:

a. Relativo.
b. Imediato.
c. Indirecto por registo com deslocamento.
d. Absoluto por registo com deslocamento.

50. A instrução virtual "li $t0, 0x10012345" da arquitectura MIPS decompõe-se na seguinte sequência de instruções nativas:

a. "lui $1, 0x2345" seguida de "ori $t0, $t1, 0x1001".

b. “ori $t0, $1, 0x1001” seguida de “ori $t0, $s1, 0x2345”.
c. “lui $1, 0x1001” seguida de “ori $t0, $t1, 0x2345”
d. “ori $t0, $1, 0x2345” seguida de “lui $1, 0x1001”.

51. A instrução virtual “bgt $t8, $t9, target” da arquitectura MIPS decompõe-se na seguinte sequência de instruções nativas:

a. “slt $1, $t8, $t9” seguida de “bne $1, $0, target”.
b. “slt $1, $t9, $t8” seguida de “bne $1, $0, target”
c. “slt $1, $t8, $t9” seguida de “beq $1, $0, target”
d. “slt $1, $t9, $t8” seguida de “beq $1, $0, target”

52. A instrução virtual bgt $8,0x16,target da arquitectura MIPS decompõe-se na seguinte sequência de instruções nativas.

a. slti $1,$8,0x17 seguida de bne $1,$0,target.
b. slti $1,$8,0x16 seguida de bne $1,$0,target.
c. slti $1,$8,0x17 seguida de beq $1,$0,target.
d. stli $1,$8,0x16 seguida de beq $1,$0,target.

53. A instrução virtual bge $t8,$t9,target da arquitectura MIPS decompõem-se na seguinte sequência de instruções nativas:

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

54. A instrução virtual ble $8,0x16, target da arquitectura MIPS decompõe-se na seguinte sequência de instruções nativas:

a. Slti $1, $8, 0x17 seguida de bne $1, $0, target.
b. Slti $1, $8, 0x16 seguida de bne $1, $0, target.
c. Slti $1, $8, 0x17 seguida de beq $1, $0, target.
d. Slti $1, $8, 0x16 seguida de beq $1, $0, target.

55. A instrução virtual div $20,$21,$22 da arquitectura MIPS decompõe-se na seguinte sequência de instruções nativas:

a. div $21,$22 seguida de mfhi $20.
b. div $21,$22 seguida de mflo $20.
c. div $20,$21 seguida de mfhi $22.
d. div $20,$21 seguida de mflo $22.

56. A instrução div $2,$1,$2 da arquitectura MIPS decompõe-se na seguinte sequência de instruções nativas:

a. Div $1,$2 seguida de mflo $2.
b. Div $1,$2 seguida de mfhi $2.
c. Div $1,$2 seguinda de mtlo $2.
d. A instrução referida já é nativa.

57. A instrução virtual la $t0,lable da arquitectura MIPS, em que lable corresponde ao segundo endereço do segmento de dados do PCSPIM, decompõem-se na seguinte sequência de instruções nativas

a. lui $1,0x1001 seguida de ori $t0,$1,0x0001.
b. ori $t0,$1,0x001 seguida de lui $1,0x1001.
c. lui $1,0x0040 seguida de ori $t0,$1,0x001.
d. ori $t0,$1,0x0001 seguida de lui $1,0x0040.

58. Considerando que $5=0xFFFFFFF7 e $10=0x00000002, o valor armazenado no registo destino pela instrução virtual rem $6, $5, $10 é:

a. $6=0x00000004.
b. $6=0xFFFFFFF4.
c. $6=0x00000001.
d. $6=0xFFFFFFFF.

59. Considerando que no endereço de memória acedido pela instrução "lb $t0, 0xFF($t1)" está armazenado o valor 0x82, o valor armazenado no registo destino no final da execução dessa instrução é:

a. 0xFF.
b. 0x82.
c. 0xFFFFFF82.
d. 0xFF82

60. Considere que no endereço de memória acedido pela instrução lb $9,0xC7 ($9) está armazenado o valor 0x83, e que no registo $9 está armazenado, antes da sua execução o valor 0x1001FF00. O valor que ficará no registo $9, no final da execução da instrução é:

a. 0x83.
b. 0x1001FFC7.
c. 0xC7.
d. 0xFFFFFF83.

61. Na arquitectura MIPS o endereço-alvo de uma instrução de salto condicional ("beq/bne") armazenada no endereço 0x00400032 cujo código original é 0x13ABFFFD é:

a. 0x00400034
b. 0x0040FFF4
c. 0x00400018
d. 0x0040001C

62. Considere uma instrução de saldo condicional residente no endereço 0x004038AC, cujo código máquina é 0x1185FFF0. O endereço-alvo dessa instrução é:

a. 0x0040386C.
b. 0x004038A0.
c. 0x00403870.
d. 0x0041389C.

63. Os endereços mínimo e máximo para os quais uma instrução "bne" presente no endereço 0x00430210 pode saltar são:

a. 0x00000000, 0xFFFFFFFF.
b. 0x00000000, 0x0FFFFFFC.

c. 0x00428214, 0x00438213.
d. 0x00410214, 0x00450210.

64. Os endereços mínimo e máximo para os quais uma instrução de salto condicional (beq ou bne) da arquitectura MIPS, presente no endereço 0x0043FFFC pode saltar são:

a. 0x0041FFFC, 0x0045FFFB.
b. 0x00437FFC, 0x00447FFB.
c. 0x00420000, 0x0045FFFC.
d. 0x00438000, 0x00447FFF.

65. Os endereços mínimo e máximo para os quais uma instrução "j" presente no endereço 0x00430210 pode saltar são:

a. 0x00428214, 0x00438213.
b. 0x00000000, 0xFFFFFFFF
c. 0x00410214, 0x00450210.
d. 0x00000000, 0x0FFFFFFC.

66. 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.

67. A instrução jal label executa sequencialmente as seguintes operações:

a. PC=PC+4, $ra=PC, PC=label.
b. PC=PC+4, PC=label, $ra=PC.
c. $ra=PC, PC=PC+4, PC=label.
d. $ra=PC, PC=label, PC=PC+4

68. A instrução "jal funct" executa sequencialmente as seguintes operações:

a. $PC = $PC + 4, $ra = $PC, $PC = funct.
b. $PC = $pc + 4, $PC = funct, $ra = $PC.
c. $ra = $PC, $PC = funct.
d. Nenhuma das anteriores.

69. A instrução jalr $5 (jump and link on register) executa sequencialmente as seguintes operações:

a. PC=PC+4, $ra=PC, PC=$5.
b. PC=PC+4, $5=PC, PC=$ra.
c. $5=PC, PC=PC+4, $ra=PC.
d. $ra=PC, PC=PC+4, PC=$5.

70. No MIPS, as instruções do tipo "I" incluem um campo imediato de 16 bits que:

a. permite armazenar um offset de endereçamento de +-32 KBytes para as instruções "sw".

b. permite armazenar um offset de endereçamento +- (32*4) KBytes para as instruções “sw”
c. permite armazenar um offset de endereçamento de +- 32KBytes para as instruções “beq”
d. permite armazenar um offset de endereçamento de +-(32*4) Kilo instruções para as instruções “beq”

71. Segundo a convenção de utilização de registos na arquitectura MIPS, uma subrotinha tem de preservar os registos

a. $s0... $s7, $v0, $v1.
b. $s0... $s7, $a0... $a3
c. $a0... $a3, $ra
d. $s0... $s7, $ra

72. Segundo a convenção de utilização de registos da arquitectura MIPS, uma subrotina não necessita de salvaguardar os registos com os prefixos:

a. $a, $v, $s.
b. $s, $v, $t.
c. $a, $v, $t.
d. $a, $s, $t.

73. O trecho de código que permite atribuir o valor 0xFF à variável “i” indirectamente através do ponteiro “p” é:

a.	b.	c.	d.

int i;
int *p;
 	
i = &p;
*i = x0FF;
	int i;
	int *p;
	
	p = *i;
	*p = 0xFF;
		int i;
		int *p;
		
		p = &i;
		*p = 0xFF;
			int i;
			int *p=0xFF;
			
			p = &I;
			I = *p;

Resposta: C.

74. Na arquitectura MIPS a stack é gerida de acordo com os seguintes princípios:

a. cresce no sentido dos endereços mais altos, apontando o registo $sp para a última posição ocupada.
b. cresce no sentido dos endereços mais baixos, apontando o registo $sp para a última posição ocupada.
c. cresce no sentido dos endereços mais altos, apontando o registo $sp para a primeira posição livre
d. cresce no sentido dos endereços mais baixos, apontando o registo $sp para a primeira posição livre.

75. Numa arquitectura MIPS a stack é gerida de acordo com os seguintes princípios:

a. cresce no sentido dos endereços mais altos, apontando o registo $sp para a última posição ocupada.

b. cresce no sentido dos endereços mais altos, apontando o registo $sp para a primeira posição livre.
c. cresce no sentido dos endereços mais baixos, apontando o registo $sp para a primeira posição livre.
d. cresce no sentido dos endereços mais baixos, apontando o registo $sp para a última posição ocupada.

76. Na convenção adoptada pela arquitectura MIPS, a realização de uma operação 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).

77. Na convenção adoptada pela arquitectura MIPS, a realização de uma operação de pop do valor do registo $ra é realizado 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. addu $sp,$sp,4 seguida de sw $ra,0($sp).
d. sw $ra, 0($sp) seguida de addu $sp,$sp,4

78. A detecção de overflow numa operação de adição de números sem sinal faz-se através:

a. da avaliação do bit mais significativo do resultado
b. do “ou” exclusivo entre o carry in e o carry out da célula de 1 bit mais significativa.
c. do “ou” exclusivo entre os 2 bits mais significativos do resultado.
d. da avaliação do carry out do bit mais significativo do resultado

79. 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.

80. Numa ALU, a detecção de overflow nas operações de adição algébrica é efectuada 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 o bit mais significativo e o menos significativo do resultado.
d. do “ou” exclusivo entre os 2 bits mais significativos do resultado.

81. Para a implementação de uma arquitectura de multiplicação de 32 bits são necessários, entre outros, registos para o multiplicador e multiplicando e uma ALU. A dimensão exacta, em bits, de cada um destes elementos deve ser:

a. Multiplicando: 32 bits; Multiplicador: 32 bits: ALU: 64 bits

b. Multiplicando: 32 bits; Multiplicador: 64 bits: ALU: 32 bits.
c. Multiplicando: 64 bits; Multiplicador: 32 bits; ALU: 32 bits.
d. Multiplicando: 64 bits; Multiplicador: 32 bits; ALU: 64 bits

82. Numa implementação de uma arquitectura de divisão de 32 bits pode recorrer-se a um algoritmo que, sem para alterar o registo que armazena o divisor:

a. Faça deslocamentos sucessivos do quociente à esquerda, mantendo o dividendo.
b. Faça deslocamentos sucessivos do quociente à esquerda e do dividendo à direita.
c. Faça deslocamentos sucessivos do quociente à esquerda e do dividendo à esquerda
d. Faça deslocamentos sucessivos do quociente à direita e do dividendo à esquerda

83. A quantidade real binária 1011,11000000(2) quando representada em decimal é igual a:

a. 12,6
b. 11,75
c. 3008,0
d. 1504,0

84. Imagina que se pretende inicializar o conteúdo do registo $f4 com a quantidade real 2.0. A sequência de instruções que efectua esta operação é:

a.
```
li      $t2, 2
mtc1    $t0, $f4
```

b.
```
lui     $t0,0x4000
mtc1    $t0, $f4
```

c.
```
li.s    $f4,2.0
cvt.s.w $f4,$f0
```

d.
```
li      $t0,2
mtc1    $t0,$f0
mov.s   $f4,$f0
```

Resposta: B.

85. O resultado da instrução multu $t0,$t1 é representável em 32 bits se:

a. HI for uma extensão com sinal de LO
b. HI=0x00000000.
c. HI for diferente de zero.
d. HI=0xFFFFFFFF.

86. Considerando que $t0= -4 e $t1= 5, o resultado da instrução mult $t0,$t1 é:

a. HI = 0x80000000, LO = 0x000000EC.
b. HI = 0xFFFFFFFF, LO = 0xFFFFFFEC.
c. HI = 0xFFFFFFEC, LO = 0xFFFFFFFF.
d. HI = 0x00000000, LO = 0xFFFFFFEC.

87. Considerando que $t0=-7 e $t1=2, o resultado da instrução div $t0,$t1 é:

a. LO=-3, HI=-1.
b. LO=-3, HI= 1.

c. LO= 1, HI=-4.
d. LO=-1, HI=-3.

88. Considerando que $t0=0x00000007 e $t1=0xFFFFFFFE, o resultado da instrução div $t0,$t1 é:

a. Hi= 0x00000001, LO=0xFFFFFFF0.
b. HI=0xFFFFFFF0, LO=0xFFFFFFFF.
c. HI=0xFFFFFFFC, LO=0x00000001.
d. Nenhuma das anteriores.

89. 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".

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

a. 0xAC838080
b. 0xAC83FF80
c. 0xAC64FF80
d. 0xAC648080

91. Considerando que $f2=0x3A600000 e $fa=0xBA600000, o resultado da instrução sub.s $f0,$f2,$f4 é:

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

92. Considere que no endereço de memória acedido pelas instruções lb $t0,0xFF($t0) e lb $t1,0xFF($t0) está armazenado o valor 0x02, o valor armazenado nos respectivos registos destino, no final da execução dessas instruções é:

a. $t0=0x000000FF, $t1=0xFFFFFFFF
b. $t0=0x00000082, $t1=0xFFFFFF82
c. $t0=0xFFFFFF82, $t1=0x00000082
d. $t0=0xFFFFFFFF, $t1=0x000000FF

93. Assumindo que o registo $f8 possui o valor (representado em hexadecimal) 0x3FE00000, após a execução da instrução cvt.d.s $f10,$f8, os registos $f10 e $f11 terão, respectivamente, os valores:

a. 0x00000000, 0x3FFC0000.
b. 0x00000000, 0x3FE00000.

c. 0x00000000, 0x07FC0000.
d. 0x07FC0000, 0x00000000.

94. Admitindo que $f8=0x00000000 e $f9=0x618A0000, após a execução da instrução cvt.s.d $f10, $f8, o registo $f10 terá o valor (assuma que $f9 contém a parte mais significativa do operando):

a. $f10=0x7F800000.
b. $f10=0x61D00000.
c. $f10=0x0C500000.
d. Nenhuma das anteriores.

95. Considere que os valores reais representados nos registos $f4 e $f6 são (em base 2) $f4=1,00011010x22 e $f6=-1,10101000x2-2. O valor armazenado no registo $f0, em hexadecimal, após a execução da instrução add.s $f0,$f4,$f6 é:

a. 0x40708000.
b. 0x40F28000.
c. 0x40650000.
d. 0xBE920000.

96. 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. -0,1625 x 21.
b. -0,1625 x 23.
c. -3,25 x 21.
d. -16,25 x 21.

97. A codificação do número +1,13125 x 101 no formato IEEE 754, precisão simples, representado em hexadecimal é:

a. 0x415A8000.
b. 0x41350000.
c. 0x3E350000.
d. 0x01DA8000.

98. A representação normalizada e arredondada para o par mais próximo de acordo com o formato IEEE 754 precisão simples do número 100,11011000000000000101102 é:

a. 1,00110110000000000000110 = 23.
b. 1,00110110000000000000110 = 2-3.
c. 1,00110110000000000000101 = 23.
d. 1,00000000000000000000000 = 23.

99. Considere duas máquinas (A e B) com implementações distintas da mesma arquitectura do conjunto de instruções (ISA). A máquina A possui uma duração de sinal de relógio de 0,5ns e a máquina B de 0,4ns. Para um dado programa a máquina A apresenta um CPI de 2,0 e a B de 3,0.

a. A máquina A é mais rápida do que a maquina B por um factor de 1,25.
b. A máquina A é mais rápida do que a maquina B por um factor de 1,2.

c. A máquina B é mais rápida do que a máquina A por um factor de 1,25.
d. A máquina B é mais rápida do que a máquina A por um factor de 1,2.

100. Numa implementação single-cycle da arquitectura MIPS:

a. 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.
b. Existem registos à saída dos elementos operativos fundamentais para guardar valores a utilizar no ciclo de relógio seguinte.
c. Todas as operações de leitura e escrita são síncronas com o sinal de relógio.
d. Existem memórias específicas para código e dados para possibilitar o acesso a ambos os tipos de informação num único ciclo de relógio.

101. A frequência de relógio de uma implementação single cycle da arquitectura MIPS:

a. É limitada pelo maior dos tempos de atraso dos elementos operativos Memória, ALU e File Register.
b. Varia em função da instrução que está a ser executada.
c. É limitada pelo maior dos atrasos cumulativos dos elementos operativos envolvidos na execução da instrução mais longa.
d. É limitada pelo menor dos tempos de atraso dos elementos operativos Memória, ALU e File Register.

102. Numa implementação single-cycle da arquitectura MIPS:

a. Existem memórias independentes para código e dados para possibilitar o acesso a ambos os tipos de informação num único ciclo relógio.
b. Existe uma única ALU para realizar todas as operações aritméticas e lógicas (incluído o cálculo do valor do PC, BTA, endereços de acesso á memoria e comparação de registos) necessários para (executar) num único ciclo de relógio qualquer uma das instruções suportadas
c. Existe uma única memoria acedida para código e e para dados em ciclos de relógio distintos.
d. Todas as operações de leitura e escrita são síncronas com o sinal de relógio

103. A unidade de controlo de uma implementação multi-cycle da arquitectura MIPS:

a. é um elemento combinatório que gera os sinais de controlo em função do campo opcode do código máquina da instrução.
b. é uma máquina de estados em que o primeiro e o segundo estados são comuns à execução de todas as instruções.
c. é uma máquina de estados com um número de estados igual ao número de fases da instrução mais longa.
d. é um elemento combinatório que gera os sinais de controlo em função do campo funct do código máquina da instrução.

104. 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:

a. calcular o valor do Branch Target Address e comparar os registos (operandos da instrução)
b. calcular o valor de PC+4 e comparar os registos (operandos da instrução).

c. comparar os registos (operandos da instrução) e calcular o valor do Branch Target Address.
d. calcular o valor de PC+4 e o valor do Branch Target Address.

105. Uma implementação pipelined de uma arquitectura possui, relativamente a uma implementação single-cycle da mesma, a vantagem de:

a. Diminuir o tempo de execução de cada uma das instruções.
b. Permitir a execução de uma nova instrução a cada novo ciclo de relógio
c. Aumentar o débito de execução das instruções.
d. Todas as anteriores.

106. A frequência de relógio de uma implementação pipelined da arquitectura MIPS:

a. É limitada pelo maior dos atrasos cumulativos dos elementos operativos envolvidos na execução da instrução mais longa.
b. É definida de forma a evitar stalls, assim como delay slots.
c. É limitada pelo menor dos tempos de atraso dos elementos operativos Memória, ALU e File Register.
d. É limitada pelo maior dos tempos de atraso dos elementos operativos Memória, ALU e File Register.

107. A técnica de forwarding/bypassing num processador MIPS pipelined permite:

a. Utilizar como operando de uma instrução um resultado produzido por outra instrução que se encontra numa etapa mais recuada do pipeline.
b. Trocar a ordem de execução das instruções de forma a resolver um hazard de dados.
c. Utilizar como operando de uma instrução um resultado produzido por outra instrução que se encontra numa etapa mais avançada do pipeline.
d. Escrever o resultado de uma instrução no File Register antes de ela chegar à etapa WB

108. Numa implementação single cycle imposta pela instrução de leitura da arquitectura MIPS, a frequência máxima de operação da memória de dados é, assumindo os atrasos a seguir indicados:

a. 32,25 MHz (T=31ns).
b. 25,00 MHz (T=40ns).
c. 29,41 MHz (T=34ns).
d. 31,25 MHz (T=32ns).

Memórias externas: leitura - 9ns, escrita – 11ns;
File register: leitura – 3ns, escrita – 4ns;
Unidade de controlo: 2ns;
ALU (qualquer operação): 7ns;
Somadores: 4ns; Outros: 0ns

109. 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 100MHz relativamente a uma single-cycle a operar a 20 MHz é de:

a. 1,25.
b. 1.

c. 5.
d. 0,8.

110. Um hazard de controlo numa implementação pipelined de um processador ocorre quando:

a. 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.
b. 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.
c. Existe uma dependência entre o resultado calculado por uma instrução e o operando usado por outra que segue mais atrás do pipeline.
d. Por azar, a unidade de controlo desconhece o opcode da instrução que se encontra na etapa ID.

111. O seguinte trecho de código, a executar sobre uma implementação pipelined da arquitectura MIPS, apresenta os seguintes hazards:

a. Um hazard de controlo na quarta instrução e um hazard de dados na segunda instrução que pode ser resolvido por forwarding.
b. Um hazard estrutural na primeira instrução e um hazard de controlo na quarta instrução.
c. Um hazard de controlo na quarta instrução e hazards de dados na segunda, terceira e na quarta instruções que podem ser resolvidos por forwarding.
d. 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, $tt2, $t0 # 3
    Beq   $t5, $t1, L1  # 4
```

112. 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:

a. lw $6,0x2020($5) na terceira fase.
b. add $4,$5,$6 na quarta fase.
c. add $4,$5,$6 na terceira fase.
d. lw $6,0x2020($5) na quinta fase.

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	Funct	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

$5	0xFF0180FF
$6	0x100100A0
$7	0x1001009C
$8	0x00001E00
$PC	0x00400048

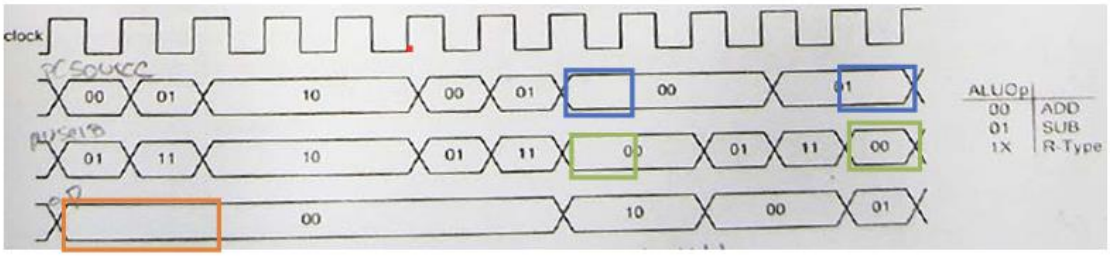
CPU

Figura 1

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

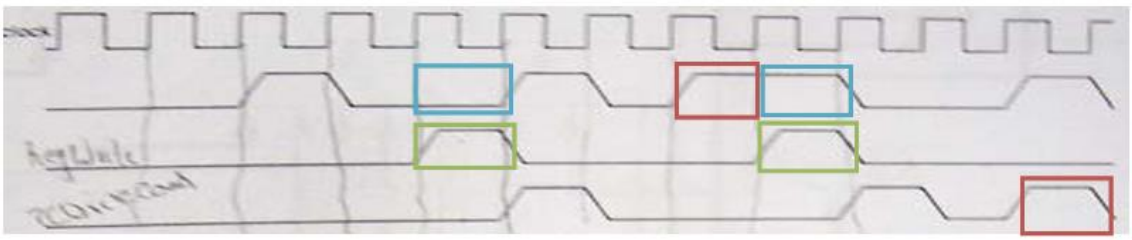
113. Para as 3 primeiras instruções do trecho de código apresentado na Figura 1, os sinais de controlo representados no seguinte diagrama temporal correspondem, pela ordem indicada, a:

a. “ALUSelB”, “ALUOp” e “PCSource”.
b. “PCSource”, “ALUOp” e “ALUSelB”.
c. “PCSource”, “ALUSelB” e “ALUOp”.
d. “ALUSelB”, “PCSource” e ”ALUOp”.



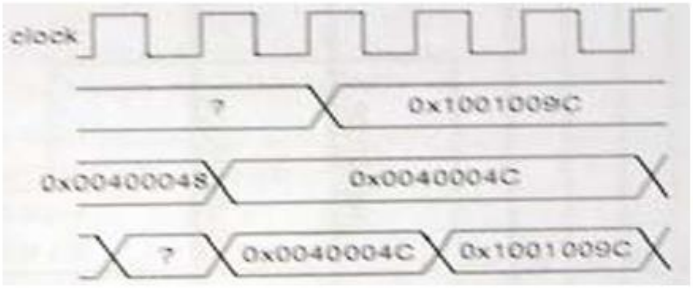
114. Também para as 3 primeiras instruções do trecho de código apresentado na Figura 1, os sinais de controlo representados no seguinte diagrama temporal correspondem, pela ordem indicada, a:

a. “RegWrite”, “PCWriteCond” e “RegDst”.
b. “RegDst”, “RegWrite” e “PCWriteCond”.
c. “PCWriteCond”, “RegWrite” e “RegDst”.
d. “RegDst”, “PCWriteCond” e “RegWrite”.



115. 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. “A”, ”InstRegister” e “PC”.
b. “B”, “PC” e “ALUOut”.
c. “A”, “PC” e “ALUOut”.
d. Nenhuma das anteriores.



116. 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 multicycle do MIPS, é (desde o instante inicial do instruction fetch da primeira instrução até ao memento em que vai iniciar-se o instruction fetch da instrução presente em “L2:”);

a. 58 ciclos de relógio.
b. 12 ciclos de relógio.
c. 6 ciclos de relógio.
d. 35 ciclos de relógio.

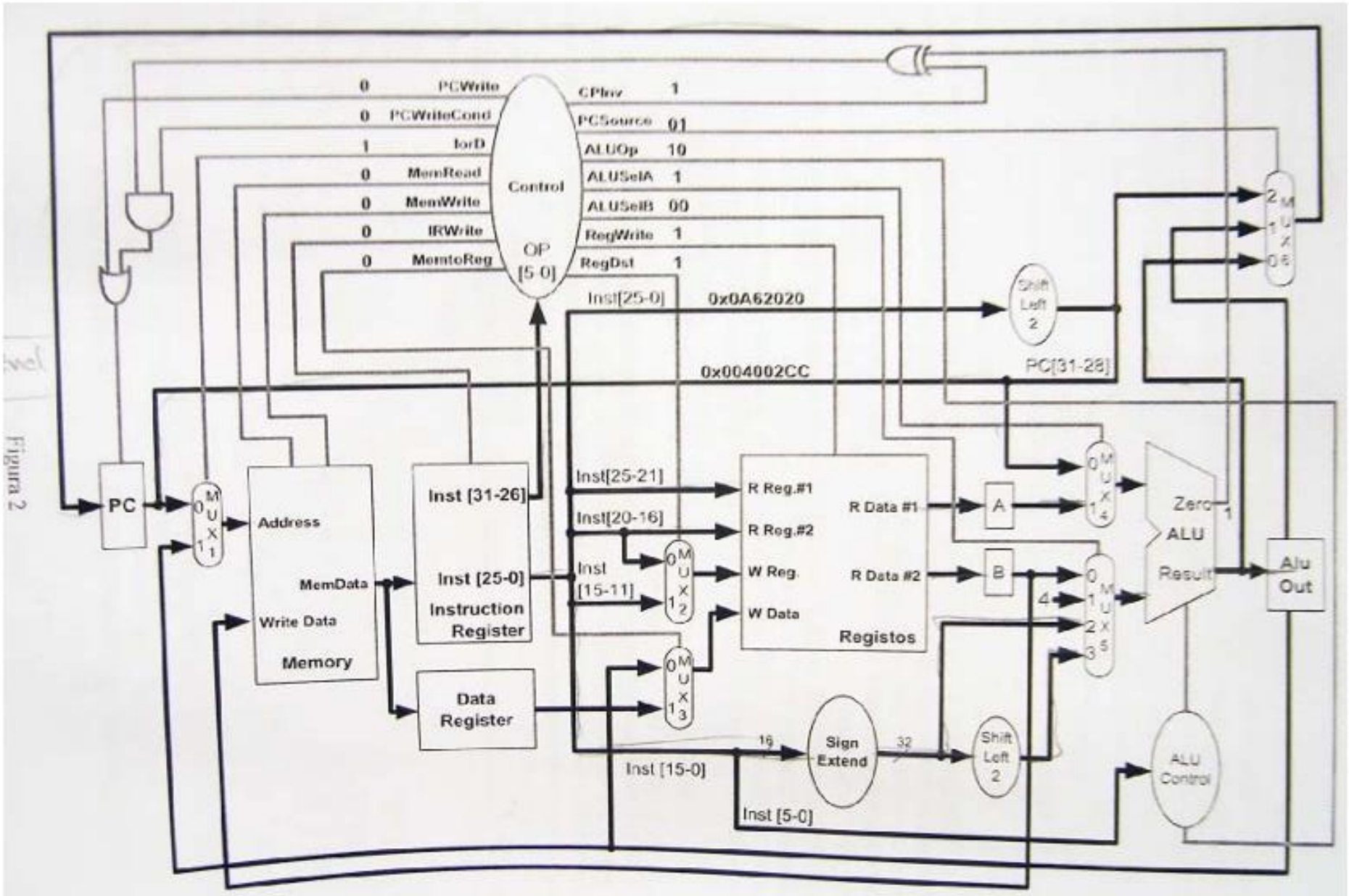

0 PCWrite
0 PCWriteCond
1 IorD
0 MemRead
0 MemWrite
0 IRWrite
0 MemtoReg
Control
OP [5-0]
CPlev 1
PCSource 01
ALUOp 10
ALUSelA 1
ALUSelB 00
RegWrite 1
RegDst 1
Inst[25-0]
0x0A62020
0x004002CC
Shift Left 2
PC[31-28]
PC
Address
MemData
Write Data
Memory
Inst [31-26]
Inst [25-0]
Instruction Register
Data Register
Inst[25-21]
Inst[20-16]
Inst [15-11]
R Reg.#1
R Reg.#2
W Reg.
W Data
R Data #1
R Data #2
Registos
A
B
4
Zero
ALU
Result
Alu Out
Inst [15-0]
16
Sign Extend
32
Shift Left 2
ALU Control
Inst [5-0]

Figura 2

117. Na arquitetura básica de um sistema computacional o Control Bus permite:

a. especificar a natureza de uma operação efetuada sobre a memória.
b. identificar, na memória, a origem/destino dos dados transferidos.
c. transferir dados entre os registos do CPU.
d. transferir o código máquina das instruções para o instruction register.

118. Na arquitetura básica de um sistema computacional o Address Bus permite:

a. transferir o código máquina das instruções para o program counter.
b. identificar, na memória, a origem/destino da informação transferida.
c. especificar a natureza das operações efetuadas sobre a memória.
d. transferir dados entre a memória externa e os registos do CPU.

119. Considere a instrução bne $6, $2, label armazenada no endereço de memória 0x00400004. Se o endereço alvo da instrução for 0x00400020, o valor dos 16 bits menos significativos do código máquina dessa instrução será:

a. 0x0018
b. 0x0007
c. 0x001C
d. 0x0006

120. Suponha que $3 = 0xFC47A6E8 e $2 = 0x00FFFF00. A instrução “xor $6, $3, $2” produz o seguinte resultado armazenado em $6:

a. 0xFC0000E8
b. 0x00B80017
c. 0x3047A9E8
d. 0xFCB859E8

121. A arquitetura MIPS não permite realizar operações lógicas e aritméticas sobre o conteúdo de registos da memória externa ou, simultaneamente, sobre o conteúdo de registos internos e de registos da memória externa. Por essa razão:

a. o número de instruções do seu set é superior ao das arquiteturas que permitem esse tipo de operações.

b. é considerada como sendo do tipo Load-Store.

c. todas as variáveis de um dado programa deverão residir em registos internos.

d. permite, num mesmo ciclo de relógio, operar sobre a ALU e efetuar uma operação de escrita na memória.

122. O ciclo de execução de uma instrução é composto pela seguinte sequência ordenada de operações:

a. Instruction fetch, operand fetch, execute, decode, store result
b. Instruction fetch, execute, decode, operand fetch, store result
c. Instruction fetch, decode, operand fetch, execute, store result
d. Operand fetch, decode, instruction feich, execute, store result

123. Os códigos ASCII dos caracteres '0' (zero) e 'a' são, respetivamente, 0x30 e 0x61. Num computador baseado num MIPS little endian a string "ac1" está armazenada a partir

do endereço 0x10010000. Sabendo que $s0=0x10010004, o valor que ficará no registo $t0 após a execução da instrução lw $t0, -4($s0) é

a. 0x61633100

b. 0x00316361

c. 0x31006361

d. 0x61630031

124. O modo de endereçamento que, na instrução lb $t0,0 ($ra), permite ler um valor da memória designa-se:

a. indireto por registo.

b. indireto por registo com deslocamento.

c. relativo ao PC.

d. pseudo-direto.

125. Considere a instrução jal sqblz, armazenada no endereço de memória 0x004004C8. O label sqblz corresponde ao endereço 0x00400518. Na conclusão da execução dessa instrução, o conteúdo do registo $ra será:

a. 0x00400518

b. 0x004004CC

c. 0x004004C8

d. 0x0040051C

126. Considere a instrução nativa or $t0, $t2, $0. Considerando a vertente funcional desta instrução, podemos dizer que a mesma se enquadra na seguinte classe de instruções:

a. instruções de controlo de fluxo de execução.

b. instruções de transferência de informação.

c. instruções de lógica bitwise.

d. instruções de processamento e alteração de informação.

127. Seja (C= A + B) com A e B codificados em complemento para dois. Quando "um" é o resultado do XOR entre o carry in e o carry out da célula de 1 bit mais significativa de C podemos afirmar que:

a. C é um resultado válido.

b. Se C é positivo então A é negativo e B é positivo.

c. Se C é negativo então A é positivo e B é negativo.

d. Se C é negativo então A e B são ambos positivos.

128. Se pretender, num programa em Assembly do MIPS, que o fluxo de execução seja desviado, de forma incondicional, para qualquer endereço múltiplo de quatro do espaço de endereçamento, deverá utilizar a instrução:

a. jr $reg

b. j label

c. jal label

d. beq $0, $0, label

129. Considerando que $1=0xFFFFFFFA e $4=0xFFFFFFFD, o conteúdo dos registos HI e LO após a execução da instrução mult $1, $4 (multiplicação signed) é:

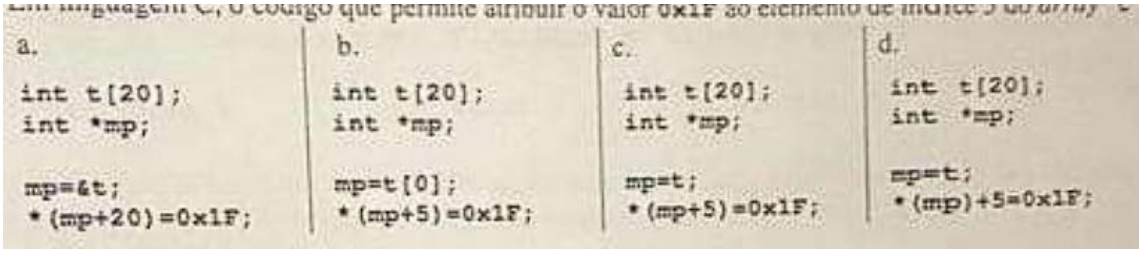
a. HI=0x00000000, LO=0x00000012

b. HI=0x00000000, LO=0xFFFFFFEE

c. HI=0xFFFFFFFF, 10=0xFFFFFFEE

d. HI=0x????????, LO=0x00000018

130. Em linguagem C, o código que permite atribuir o valor 0x1F ao elemento de índice 5 do array "t" é:



Resposta C

131. Considere um sistema baseado numa arquitetura em que o respetivo ISA especifica uma organização de memória do tipo byte-addressable. Se uma variável de 64 bits, usada num dado programa, se encontrar armazenada em memória a partir do endereço 0x25A18, pode concluir-se que a dimensão mínima do espaço de endereçamento dessa arquitetura é:

a. 2^32 endereços.

b. 2^16 endereços.

c. 2^6 endereços.

d. 2^18 endereços.

132. No MIPS, uma instrução de salto condicional:

a. é codificada usando o formato de codificação I.

b. é codificada usando o formato de codificação J.

c. é codificada usando o formato de codificação R.

d. é sempre virtual.

133. Quando, no decurso da execução de uma instrução do MIPS, a informação se obtém acedendo ao registo codificado no campo rt do código máquina dessa instrução, diz-se que estamos perante um endereçamento:

a. imediato.

b. tipo registo.

c. pseudo-direto por registo.

d. indireto por registo com deslocamento.

134. Considere uma instrução de salto incondicional (j) da arquitetura MIPS, armazenada no endereço 0x0043FFFC Os endereços mínimo e máximo para os quais esta instrução pode saltar são:

a. 0x00000000, 0xFFFFFFFF

b. 0x00420000, 0x0045FFFC.

c. 0x00000000, 0x0FFFFFFC.

d. 0x0041FFFC, 0x0045FFF8.

135. Na arquitetura MIPS a stack é gerida de acordo com os seguintes regras:

a. cresce no sentido descendente dos endereços, apontando o registo $sp para a última posição ocupada.

b. cresce no sentido descendente dos endereços, apontando o registo $sp para a primeira posição livre.

c. cresce no sentido ascendente dos endereços, apontando o registo $sp para a última posição ocupada.

d. cresce no sentido ascendente dos endereços, apontando o registo $sp para a primeira posição livre.

136. Considere que o segmento de dados de um programa contem as seguintes diretivas:

L1: .word 0, 0, 0x2345

L2: .asciiz “TeoricaPrimeira”

.align 3

L3: .space 16

L4:

Se o endereço correspondente ao label L1 for 0x10010A18, o endereço a que corresponde o label L4 é:

a. 0x10010A38

b. 0x10010A3C

c. 0x10010A48

d. 0x10010A40

137. Qual a instrução virtual do MIPS que é implementada pela seguinte sequência de instruções nativas:

xori $1, $0,0x59

altu $1, $1, $5

beq $1, $0, L1

a. bitu $5,0x59, L1

b. bgt $5,0x59,L1

c. bleu $5, 0x59, L1

d. bge $5,0x59, L1

138. Admita que no endereço ex00400068 se encontra armazenada a instrução cujo código máquina é 0x8DB6FFFC (pode estar mal o FFFC ta bem de certeza) O código assembly correspondente a esta instrução do MIPS será:

a. lw $22, -0xFFFC ($13)

b. lw $13, -0x12 ($22)

c. lw $13, -12 ($22)

d. lw $22, -4 ($13)

139. Admita que no endereço 0x0040007C se encontra armazenada a instrução cujo código máquina é 0x00400823. O código assembly correspondente a esta instrução do MIPS será:

a. sra $2,$0,12

b. subu $1, $2,$0

c. sra $1, $1,17

d. subu $2,$0, $1

140. Considere o seguinte segmento de código assembly:

lui $3, 0xC614

ori $4, $3, 0x19A5

sw $4, 0x40 ($6)

lbu $8, 0x42 ($6)

No final da execução do código anterior, o valor de $8, num MIPS little endian, é:

a. 0x00000014
b. 0x00000019
c. 0x000000A5
d. 0X000000C6

141. Considerando que $t0=0x00000022 c $t1=0xFFFFFFF6, o resultado da instrução div $to, $t1 é:

a. HI=0xFFFFFFFD, LO-0x00000003.

b. HI-0x00000002, LO 0xFFFFFFFE.

C. HI 0xFFFFFFFD, LO=0xFFFFFFF6.

d. HI-0x00000004, LO=0xFFFFFFFD.

142. O resultado da instrução mult $t0, $t1 é representável em 32 bits se:

a. HI for diferente de zero.

b. HI=0xFFFFFFFF.

c. HI for uma extensão com sinal de LO.

d. HI=0x00000000.

143. Os processadores da arquitetura hipotética ACITEO 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). Pode concluir-se, portanto:

a. que o número de registos internos é 8 e o campo opcode tem 8 bits.

b. que o número de registos internos é 32 e o campo opcode tem 6 bits.

c. que o número de registos internos é 16 e o campo opcode tem 6 bits.

d. que o número de registos internos é 16 e o campo opcode tem 7 bits.

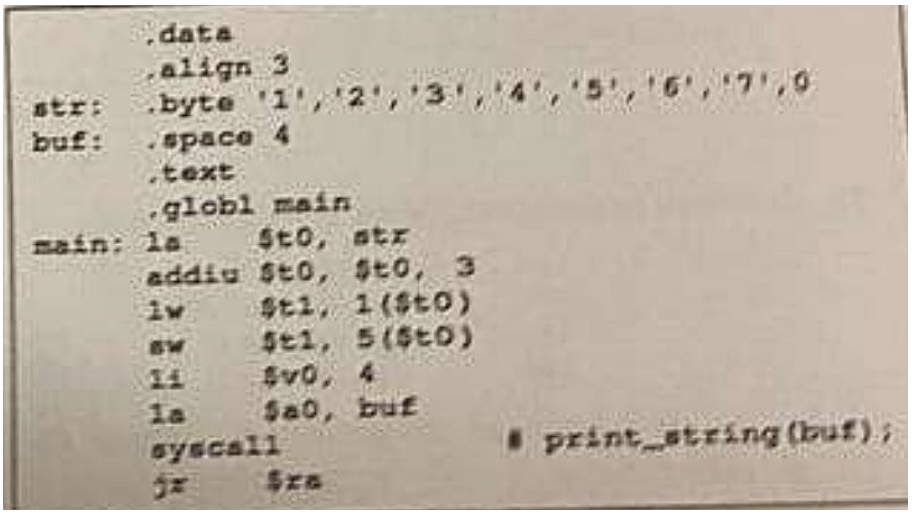
144. Ao executar o trecho de código da figura ao lado:

a. será apresentada no ecrã a string "567".

b. será gerada de uma exceção na instrução "lw" devido a uma tentativa de acesso não alinhado à memória.

c. será apresentada no ecrã a string "3456".

d. não é possível saber o resultado da instrução syscall uma vez que o espaço de memória que começa no label "but" não é inicializado.



```
        .data
        .align 3
str:    .byte '1','2','3','4','5','6','7',0
buf:    .space 4
        .text
        .globl main
main:   la    $t0, str
        addiu $t0, $t0, 3
        lw    $t1, 1($t0)
        sw    $t1, 5($t0)
        li    $v0, 4
        la    $a0, buf
        syscall                 # print_string(buf);
        jr    $ra
```

145. No decurso da execução de uma instrução sobre uma implementação single cycle da arquitetura MIPS, o valor presente na saída do registo PC corresponde:

a. ao endereço de retorno das funções.

b. ao endereço da instrução em curso.

c. ao endereço da instrução seguinte.

d. ao endereço alvo se a instrução for um j.

146. Considere uma arquitetura von Neumann, com um barramento de endereços de 24 bits e uma organização de memória byte-addressable. A dimensão máxima, expressa em bytes, que um programa a executar neste sistema (instruções + dados + stack) poderá ter, será:

a. 16 MByte

b. 256 MByte

c. 64 MByte

d. 1 GByte

147. No formato de vírgula flutuante IEEE 754, precisão dupla, um expoente positivo é codificado:

a. com um valor superior a 0 e inferior a 128.

b. com um valor superior a 0 e inferior a 1024.

c. com um valor superior a 1023 e inferior a 2047.

d. com um valor que tem o bit mais significativo a zero.

148. Numa implementação multy-cycle de um processador MIPS, o sinal MemRead, quando ativo, habilita a leitura da memória:

a. de forma assíncrona e imediata.

b. de forma assíncrona no próximo ciclo do sinal de relógio.

c. no próximo flanco ativo do sinal de relógio.

d. durante o tempo correspondente a 1 ciclo de relógio.

149. Uma arquitetura hipotética baseada num datapath single cycle em que os operandos das instruções aritméticas e lógicas podem residir em registos internos ou na memória externa, pode ser classificada como:

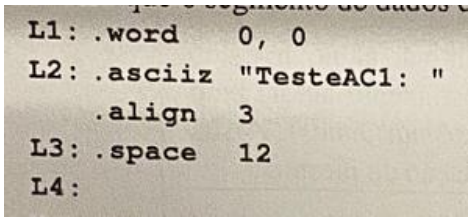
a. uma arquitetura Harvard do tipo "Load-store".

b. uma arquitetura Von Neumann do tipo "Register-Memory".

c. uma arquitetura Von Neumann do tipo "Load-store".

d. uma arquitetura Harvard do tipo "Register-Memory".

150. Considere que o segmento de dados de um programa contém as seguintes diretivas:



```
L1: .word   0, 0
L2: .asciiz "TesteAC1: "
    .align  3
L3: .space  12
L4:
```

Se o endereço correspondente ao label L1 for 0x10010A0C, o endereço do label L4 é:

a. 0x10010A24

b. 0x10010A40

c. 0x10010A3C

d. 0x10010A2C

151. Suponha que um datapath single cycle está a executar a instrução add $3, $4, $5. As operações que serão realizadas na próxima transição ativa do sinal de relógio são:

a. conclusão da operação na ALU e escrita no banco de registos.

b. escrita no banco de registos e atualização do estado da unidade de controlo.

c. escrita no banco de registos e escrita no PC.

d. escrita no banco de registos no endereço que consta do campo rt.

152. O valor à saída da ALU, na conclusão do segundo ciclo de relógio num datapath multi-cycle, durante a execução da instrução com o código máquina 0x10430024, supondo que o valor à saída do registo Program Counter é 0x00400034, é:

a. 0x00400058

b. 0x004000C8

c. 0x004000C4

d. 0x0040005C

153. Numa implementação single cycle da arquitetura MIPS, a frequência máxima de operação é de 2GHz (para os atrasos de propagação a seguir indicados). O atraso máximo que pode ocorrer nas operações da ALU será:

Memórias externas: leitura - 175ps, escrita - 150ps;

File register: leitura - 25ps, Escrita - 35ps;

Unidade de Controlo: 10ps; Somadores: 50ps;

Outros: 0ns; Escrita noutros elementos de estado: 20ps.

a. teria de ser inferior a 0ps

b. 80ps

c. 90ps

d. 110ps

154. Na implementação multi-cycle de um processador MIPS, na execução da instrução xor $1, $2, $3, a unidade de controlo ativa o sinal:

a. MemWrite no primeiro ciclo de relógio.

b. PCWrite no segundo ciclo de relógio.

C. RegWrite no quarto ciclo de relógio.

d. IRWrite no segundo ciclo de relógio.

155. Suponha a mantissa com o valor binário 1.00111000. Admitindo que a parte fracionária da mantissa deve ser armazenada em 4 bits, o valor final da mantissa após arredondamento para o par mais próximo será:

a. 1.0010

b. 1.0100

c. 1.0011

d. 1.0101

156. Numa memória com uma organização do tipo word-addressable:

a. cada posição de memória permite o armazenamento de 1 byte de informação.

b. para o mesmo espaço de endereçamento, é possível armazenar mais informação do que numa memória byte-addressable.

c. cada posição de memória é identificada com um endereço de 32 bits.

d. o acesso apenas pode ser efetuado por instruções que transferem 1 byte de informação.

157. Os processadores da arquitetura hipotética A* implementam um total de 74 instruções e têm 16 registos internos. Todas as instruções são codificadas com 32 bits, e um dos formatos tem 4 campos: opcode, dois campos para identificar registos internos e um campo para codificar constantes. Considerando que o número de bits do campo opcode é apenas o necessário para codificar as 74 instruções, conclui-se que a constante:

a. pode tomar valores na gama [0, 65535] se codificada como inteiro sem sinal.

b. pode tomar valores na gama [-16384, +16383] se codificada em complemento para dois.

c. pode tomar valores na gama [-32768, +32767] se codificada em complemento para dois.

d. pode tomar valores na gama [-65536, +65535] se codificada em complemento para dois.

158. Num datapath multi-cycle, o fator que limita a frequência máxima do relógio do CPU, é:

a. o maior dos tempos envolvidos no acesso/operação sobre os elementos memória, banco de registos e ALU.

b. o tempo de write back na instrução lw.

c. o tempo total necessário para realizar a instrução lw.

d. o tempo de transição mínimo entre estados, na máquina de estados da unidade de controle.

159. No terceiro ciclo de relógio de execução da instrução lw numa arquitetura MIPS multi-cycle é calculada na ALU a soma do conteúdo do:

a. registo codificado no campo RS da instrução com o valor do offset estendido com zeros para 32 bits.

b. registo codificado no campo RS da instrução com o valor do offset estendido com sinal para 32 bits.

c. registo codificado no campo RT da instrução com o valor do offset estendido com sinal para 32 bits.

d. program counter com o valor do offset estendido com sinal para 32 bits e multiplicado por 4.

160. A unidade de forwarding de uma implementação pipelined da arquitetura MIPS é um bloco:

a. combinatório responsável pelo avanço das instruções no pipeline.

b. combinatório que deteta a dependência entre instruções que se encontram em execução no pipeline e que gera os sinais de controlo que permitem a uma dada instrução a utilização de resultados produzidos por instruções que se encontram em estágios de execução mais avançados.

c. combinatório responsável pela geração dos sinais de controlo que permitem o encaminhamento de um resultado para estágios mais avançados do pipeline.

d. baseado numa máquina de estados que deteta situações em que uma dada instrução pode avançar, num ciclo de relógio, mais do que um estágio no pipeline.

161. As instruções virtuais assembly do MIPS (que não existem no respetivo ISA) são convertidas pelo Assembler:

a. em duas instruções nativas do MIPS.

b. num número variável de instruções nativas do MIPS que depende da instrução virtual em causa.

c. numa instrução nativa do MIPS.

d. em uma ou duas instruções nativas do MIPS.

162. Os processadores da arquitetura hipotética NMFEC possuem 16 registos internos e todas as instruções são codificadas em 28 bits. Num dos formatos de codificação existem 5 campos: um campo para o opcode, três campos para identificar registos internos em operações aritméticas e lógicas e um campo para codificar valores constantes imediatos em complemento para dois na gama [-1024 a +1023]. O número de instruções distintas que podem, no máximo, ser implementadas nesta arquitetura, será:

a. 128 instruções

b. 16 instruções

c. 64 instruções

d. 32 instruções

163. A instrução do MIPS presente no endereço 0x1040000C, a que corresponde o código máquina 0x081000F2 é:

a. j 0x001000F2

b. j 0x101000F2

c. j 0x004003C8

d. j 0x104003C8

164. A representação da quantidade -362, $5_{10} \times 10^{-2}$ no formato IEEE 754 precisão simples, é:

a. 0xC0680000
b. 0xBF680000
c. 0x3F680000
d. 0xC0740000

165. Considerando que $f4=0xBE100000 e $f6=0xC0800000, o resultado armazenado em $f8 pela instrução mul.s $f8, $f4, $f6 é:

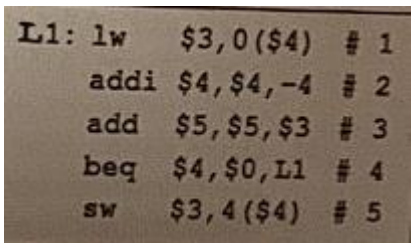
a. $f8=0x3F480000

b. $f8=0xBF200000

c. $f8=0x3F100000

d. $f8=0xC0100000

166. O segmento de código apresentado, a executar sobre uma implementação pipelined da arquitetura MIPS com unidade de forwarding para os estágios EX e ID, apresenta os seguintes hazards:



a. Um hazard de controlo na quarta instrução e hazards de dados na terceira e quarta instruções que podem ser resolvidos por forwarding para EX.

b. Um hazard de controlo na quarta instrução, um hazard de dados na terceira instrução que pode ser resolvido por forwarding para EX e um hazard de dados na quarta instrução que pode ser resolvido por forwarding para ID.

c. Um hazard de controlo na quarta instrução e hazards de dados nas segunda, terceira e quarta instruções que podem ser resolvidos por forwarding para EX.

d. Um hazard estrutural na primeira instrução, um hazard de controlo na quinta instrução, um hazard de dados na segunda instrução que pode ser resolvido por forwarding para ID e hazards de dados nas terceira e quarta instruções que podem ser resolvidos por forwarding para EX.

167. Considere os seguintes valores de elementos de estado do datapath da Figura 1, na segunda fase de execução da instrução lw $7,-4 ($5): PC=0x00400030, $5=0x00001A28, $7=0x00003C24. Nesse cenário, o valor disponível à saída do registo ALUOut durante a terceira fase de execução dessa instrução é:

a. 0x00400020

b. 0x00001A24

c. 0x00400024

d. 0x0040002C

168. Suponha que, no datapath da Figura 1, imediatamente antes da ocorrência da transição ativa do sinal de relógio, os sinais de controlo PCWrite, ALUOP, RegDst e PCSource têm os valores '1', "00", '1'e "00", respetivamente. Pode então concluir-se que:

a. está em execução uma instrução de salto incondicional na terceira fase.

b. está em execução uma instrução tipo R na terceira fase.

c. está em execução uma instrução de salto condicional na terceira fase.

d. está em execução uma instrução na primeira fase.

169. Considere uma implementação pipelined da arquitetura MIPS. O hazard de dados existente na sequência de instruções "lw $1, 0 ($2)" seguida de "beq $1, $3, target" pode ser resolvido:

a. com stall durante 2 ciclos de relógio.

b. sem recurso a stalling, com forwarding de EX/MEM para EX.

c. com stall durante 1 ciclo de relógio seguido de forwarding de MEM/WB para EX.

d. com stall durante 1 ciclo de relógio seguido de forwarding de EX/MEM para ID.

170. Tome como referência as tabelas a seguir apresentadas. Admita que o valor presente no registo PC é 0x0040005C e corresponde ao endereço da primeira instrução do programa. Considere ainda a implementação pipelined da arquitetura MIPS que estudou nas aulas, com delayed-branch e forwarding para EX (MEM/WB-EX, EX/MEM→EX) e para ID (EX/MEM→ID).

Endereço	Dados
...	...
0x684C	0x093B863D
0x6848	0x14A0C373
0x6844	0xC31748FE
0x6840	0x601F3212
0x683C	0x26B51E8C
0x6838	0x0F193ABA
0x6834	0x0B506C98
0x6830	0x03C12972

```
L0:  addi $4,$0,0x0005
     xor  $2,$2,$2
     addi $3,$0,0x684C
L1:  lw   $5,0($3)
     and  $6,$5,$4
     beq  $6,$0,L2
     add  $2,$2,$5
     j    L1
     addi $3,$3,-8
L2:  sw   $2,-64($3)
```

A execução completa do trecho de código fornecido, desde o instruction fetch da instrução referenciada pelo label LO até à conclusão da instrução referenciada pelo label L2, demora:

a. 14 ciclos de relógio.

b. 38 ciclos de relógio.

c. 54 ciclos de relógio.

d. 42 ciclos de relógio.

171.Na situação apresentada na questão anterior, 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 quinto ciclo de relógio, contado a partir do instante zero, é:

a. 0x00000000

b. 0x0000684C

c. 0x00000005

d. 0x093B863D