TESTE AC1 2018/2019

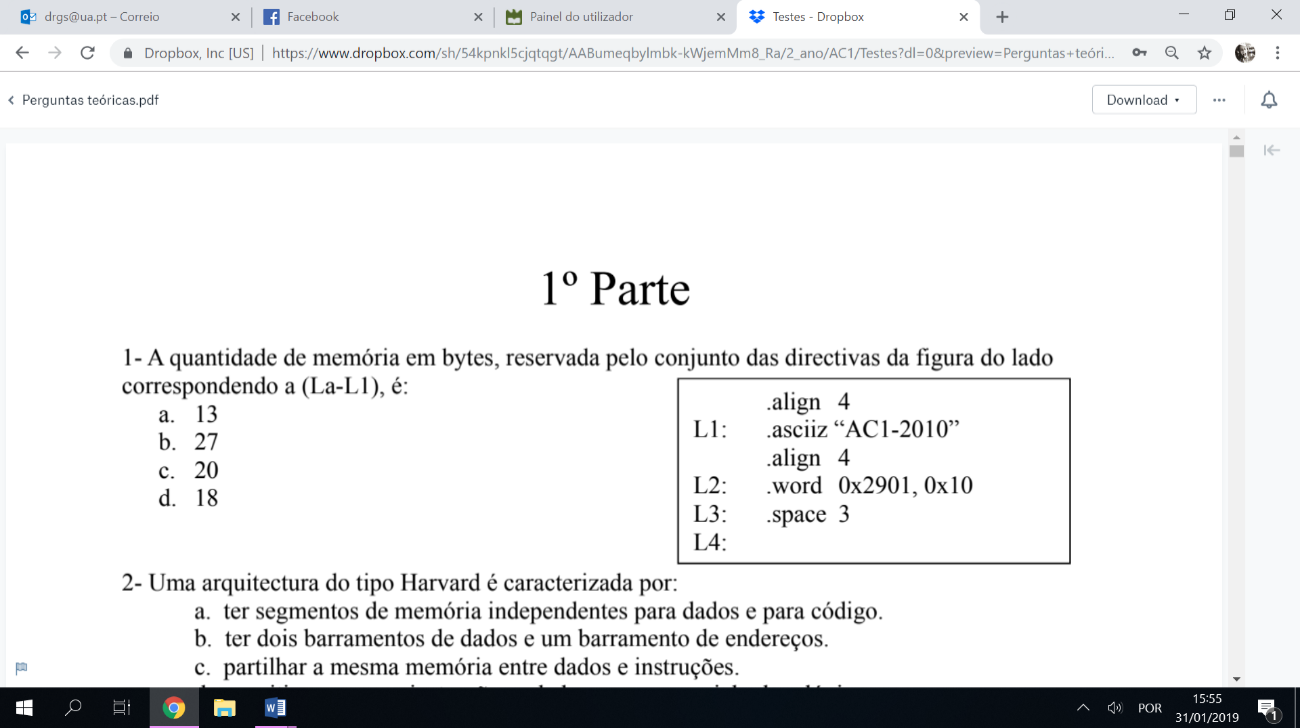
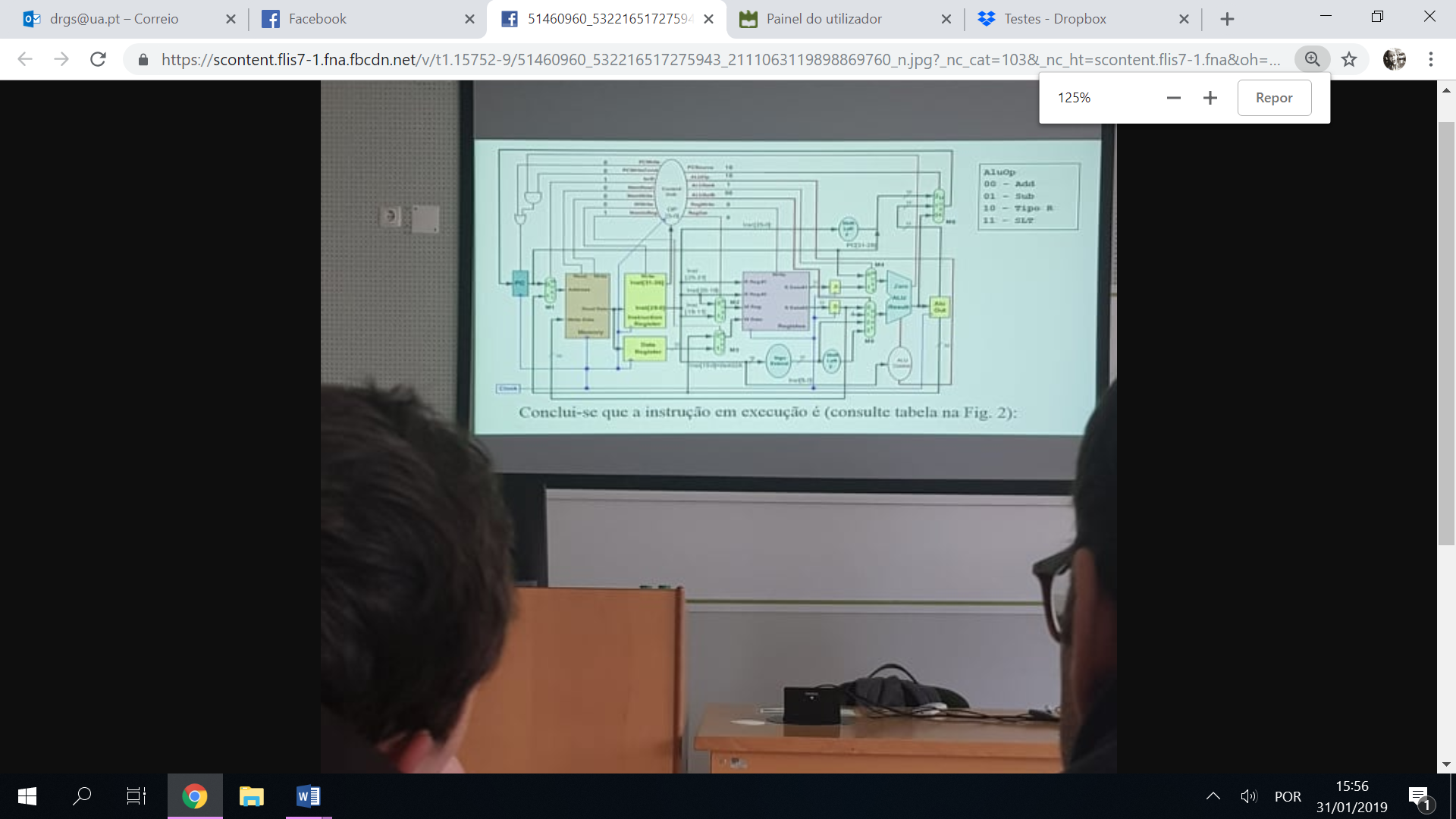
1. Na arquitetura de um sistema computacional, o ISA (arquitetura do set de instruções) define o que:
2. Um endereço de memoria externa num sistema computacional é:
3. Numa arquitetura básica num sistema computacional o “*Control Bus*” permite:
4. Numa operação de adição 2 quantidades de 16 bits. Codificação sem sinal, quando o *carryout* do resultado é 1:
5. Numa arquitetura do tipo “Register-Memory”:
6. Um datapath single-cycle adota arquitetura do tipo:
7. O modo de endereçamento usado na instrução lui $1, 0x33BC da arquitetura MIPS é do tipo:
8. Considere e indique qual destas usa endereços pseudo-direto
9. Na convenção adotada pela arquitetura MIPS, a realização de uma operação de leitura de uma word armazenada no topo da stack para o registo $ra (com atualização do registo $sp) é realizada pela seguinte sequencia de instruções:
10. A instrução addiu $2,0,0xC0F1 coloca qual valor no registo destino:
11. O resultado mult $5, $6, é representável em 32 bits se o registo HI:
12. Mulu $6, $2, $3 decompõe-se no seguinte sequencia de instrução nativas:
13. O formato I inclui um campo imediato de 16 bits que permite codificar:
14. Na arquitetura MIPS, uma instrução de jump pode saltar para qualquer endereço múltiplo de 4:
15. Considere uma arquitetura em que existe uma memoria do tipo ROM onde se encontram armazenados vários programas e uma memoria do tipo RAM onde se encontram armazenados dados. Sabendo-se que o espaço de endereçamento é comum às duas memórias pode afirmar-se:
16. Considere uma instrução beq $2, $3, label armazenada no endereço 0x0043FF00. Se o endereço alvo da instrução (label) for 0x0043FEF4, o valor dos 16 bits menos significativos do código máquina desta instrução será:
17. Numa implementação single-cycle da arquitetura MIPS, a máxima frequência de relógio a que o circuito pode funcionar tem um período que é limitado:
18. No segundo ciclo de relógio da execução de uma instrução de salto incondicional numa arquitetura MIPS multi-cycle são realizadas as seguintes operações:
19. Uma implementação multi-cycle da arquitetura MIPS, igual à dada nas aulas, possui, relativamente a uma implementação single-cycle da mesma, a vantagem de:
20. Uma implementação pipelined de uma arquitetura possui, relativamente a uma implementação multi-cycle da mesma, a vantagem de:
21. O trecho de código que permite decrementar a variável “k” indiretamente através do ponteiro “p” é:
22. Char k; b) char k; c) char k; d) char k;

Chara\*p; char \*p; char \*p; char \*p;

… … … …

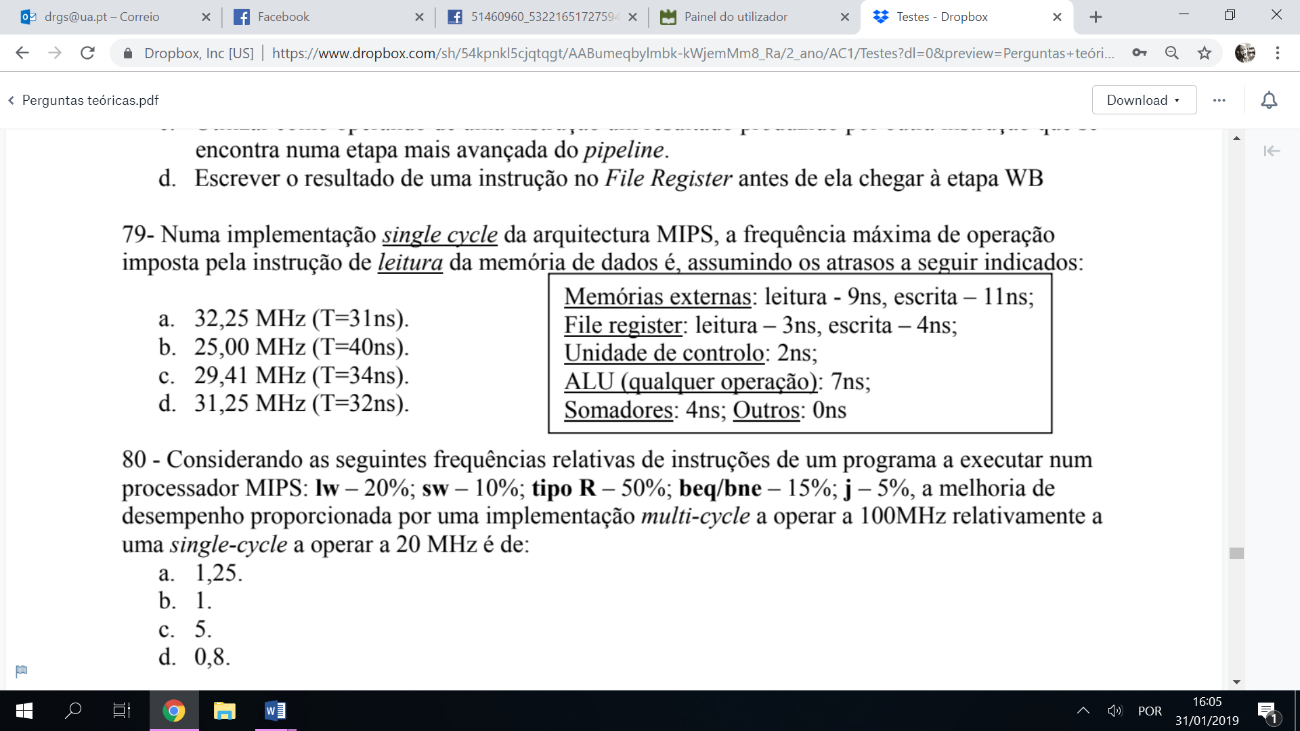
k=&p; p=&k; p=&k; \*k=p;

(\*p)--; (\*p)--; \*(p--); \*(p--);

1. Pretende-se arredondar a quantidade binaria 1.01010100 mantendo apenas 4 bits na sua parte fracionaria. Usando a técnica de arredondamento para o ímpar mais próximo, o resultado desse arredondamento é:
2. Numa implementação multi-cycle de um processador MIPS, o sinal MemWrite, quando ativo, habilita a escrita:
3. A quantidade de memoria, em bytes, reservada pelo conjunto das diretivas da figura ao lado, correspondendo a (L4-L1), é:
4.  IorD =1; MemtoReg =1; PCsource= 10; AluOp=10;AluSelA=1

(resto tudo 0)

1. Os processadores da arquitetura hipotética Xion19 implementam um total de 128 instruções. Todas as instruções são codificadas em 32 bits, num formato com 5 campos: opcode, 3 campos para identificar registos internos e um campo para codificar valores imediatos na gama [-4096,+4095]. Conclui-se, portanto, que:



2. Considerando que $4 = 0XFFFFFFF0 e $6= 0x00000005, o valor armazenado nos registos HI e LO apos a execução da instrução “div $4, $6” (divisão signed) é:
3. O jump target address de uma instrução armazenada no endereço 0x004002CB com código máquina 0x08CE0611 é:
4. A representação da quantidade -7, 375(10), no formato IEEE 754 precisão simples, é:
5. O seguinte trecho de código, a executar sobre uma implementação pipeline da arquitetura MIPS com delayed branches, e unidade de forwarding de EX/MEM e MEM/WB para o estágio EX, apresenta os seguintes hazards:

Label: lw $t3, 0($t4) #1

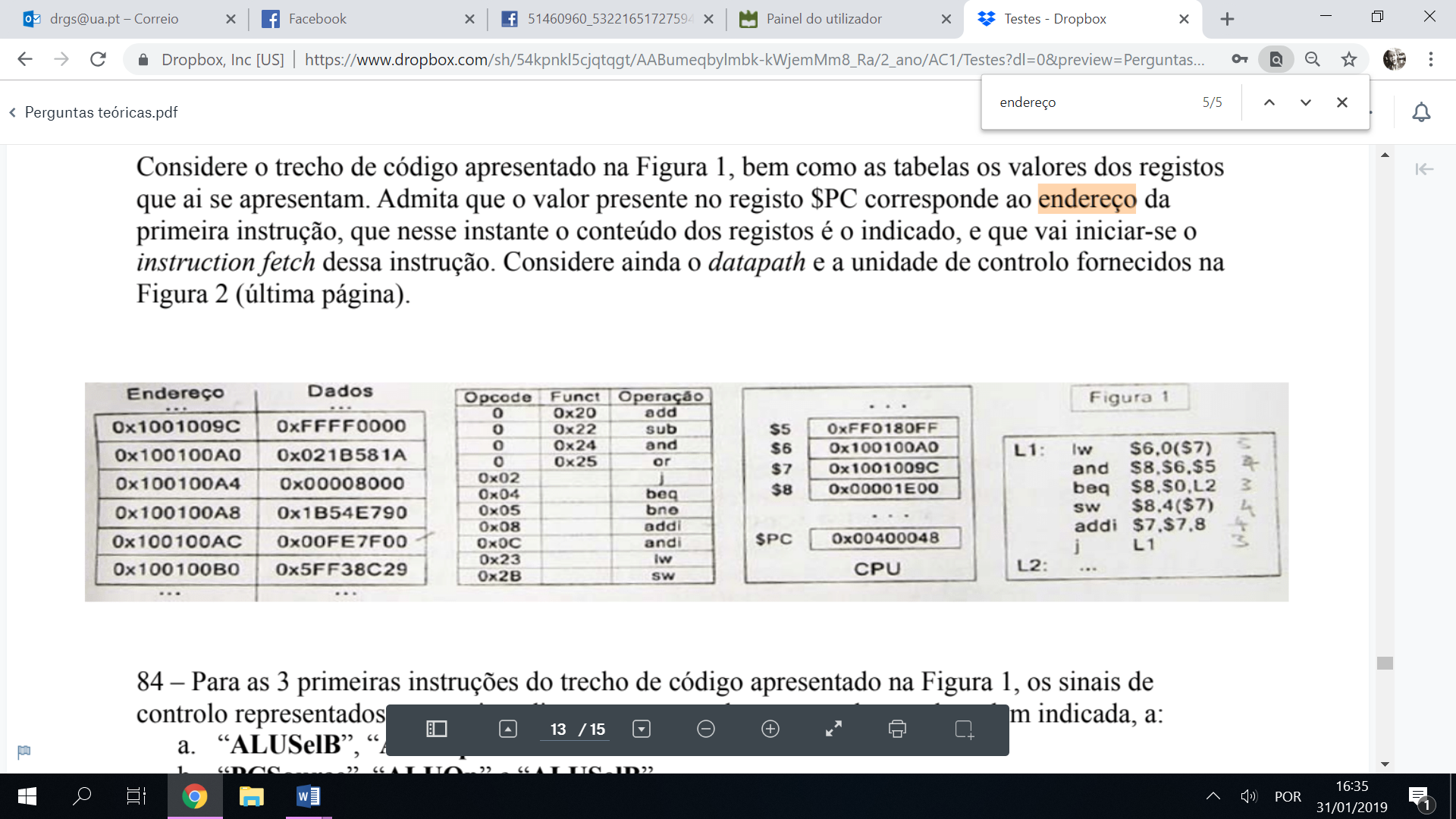
Sub $t7, $t5, $t6 #2

Ori $t2, $0, $0 #3

Beq $t2, $0, Label #4

Add $t4, $t2, $t7 #5

1. Se se pretendesse incluir suporte para a execução da instrução JR no datapath da Figura 1, umas das alterações a efetuar seria:
2. Admita que o valor presente no registo $PC corresponde ao endereço do label L1, que nesse instante o conteúdo dos registos é o indicado, e que vai iniciar-se o Instruction Fetch dessa instrução. Considere ainda a implementação pipelined da arquitetura MIPS que estudou nas aulas com delayed-branch slot e forwarding para EX e para ID.

Endereço Dados

|  |  |
| --- | --- |
| 0x5700 | 0x31303220 |
| 0x5704 | 0x31434120 |
| 0x5708 | 0x00000032 |
| 0x570C | 0xE0DE0AC1 |
| 0x5710 | 0X00000300 |
| 0x5714 | 0x000000FF |

CPU L1: addi $3, $0, 0x5700

$2 – 0x00000000 L2: addi $2, $2, 1

$7 - 0x00000001 lw $4, 0($3)

$PC – 0x004000C4 lã $5, 4($3)

Slt $1, $4, $5

Bel $1, $7, L2

Addi $3, $3, 8

Sw $5, 0($3)

L3: xor $0, $0, $0

A execução completa do trecho de código fornecido, desde o intruction fetch da instrução referenciada pelo label L1 ate á conclusão da instrução referenciada pelo label L3, demora:

34 – Quando a instrução “lw $4, 0($ 3)” esta na sua fase EX de execução, o valor armazenado no registo $PC é:

35 – Considere o seguinte trecho de código fornecido na tabela da figura 2:

Slt $1, $4, $5

Beq $1, $7, L2

a)0x0085082A 0x1027FFEC

b)0xA8850800 0x1027FFEC

c) 0x0085082A 0x1027FFFB

d) 0xA8850800 0x1027FFEC