Critérios de correção:

Grupo I

Pergunta 1. a) - 1,5 valores

AB\CD	00	01	11	10
00	0	**1**	**1**	X
01	0	**1**	**1**	0
11	**1**	0	0	**1**
10	**X**	0	X	**1**

/AD+A/D

Pergunta 1. b) - 0,5 valores -- Exame

AB\CD	00	01	11	10
00	**0**	1	1	**X**
01	**0**	1	1	**0**
11	1	**0**	**0**	1
10	X	**0**	**X**	1

(**A+D**)(/A+/D)

Critérios (positivos contam de 0 para cima, negativos são penalizações contando de cima para baixo):

- 0,3 - mapa correctamente construído
- -0,3 - um ou mais laços não máximos
- -0,3 - utiliza mais laços que os necessários
- -0,3 - um ou mais laços não correctamente convertidos em expressões lógicas
- -0,3 - não existe relação identificada entre laços e a respectivo termo na expressão lógica

Pergunta 2. a) - 0,5 valores

ABCh 1010.1011.1100b 101.010.111.100b 5274(8)

- 0,2 - conversão para binário
- 0,3 - conversão para octal

Pergunta 2. b) - 0,5 valores -- Exame

723(8) 111.010.011b 1+2+16+64+128+256 467

- 0,2 - conversão para binário
- 0,3 - conversão para decimal

- 0,5 - conversão com a fórmula do número (dígito multiplicado pela base elevado à posição)

Pergunta 3. a) - 1 valor

-101 101(10): 01100101b complemento: 10011010b +1: 110011011b

- 0,3 - conversão para binário
- 0,4 - complemento (8 bits)
- 0,3 - soma de uma unidade
- 0,1 - valor incorreto mas binário de 8 bits, sem qualquer outras contas

Pergunta 3. b) - 1 valor -- Exame

1110110b 111,0101b 7 + 0,25 + 0,0625 7,3125

- 0,5 conversão inteira
- 0,5 conversão decimal
- -0,1 erro de contas

Grupo II

Pergunta 1. - 1,5 valores

1. (A+C)(ABC+A/BC)+BC
2. (A+C)AC(B+/B)+BC
3. (A+C)AC+BC
4. AC+BC
5. (A+B)C

- -0,3 primeiro passo incorreto, existindo vários passos corretos
- -0,3 dois ou mais passos incorretos, existindo vários passos corretos
- 0,3 um passo correto
- -0,3 expressão final pode ser simplificada
- -0,3 passo correcto, mas sem regra existente (ou múltiplos passos realizados de uma só vez)

Pergunta 2. - 1 valor -- Exame

Podiam partir de duas expressões:
(A+B)C /(/(A+B)+/C) /(/A/B+/C) /(/A/B)//C //(/(/A/B)//C)

AC+BC //(AC+BC) /(/(AC)/(BC))

/A não é preciso converter para /(AA)

- 0,5 expressão lógica tem apenas portas NAND
- 0,5 circuito desenhado corretamente

Pergunta 3. - 1 valor -- Exame

AC+BC //(AC+BC) //(//(AC)+//(BC)) //(/(/A+/C)+/(/B/C))

(A+B)C /(/(A+B)+/C)

- 0,5 expressão lógica tem apenas portas NOR
- 0,5 circuito desenhado corretamente

Pergunta 2/4. - 1,5 valores

00 - 0; 10/01/11 - C --- pergunta avaliada de acordo com a resposta dada na alínea 1

- 0,5 - por uma ou mais das entradas corretas, e outras entradas incorretas

Grupo III

Pergunta 1. - 2 valores

Entrada	Estado atual	Estado seguinte	Saída
00	00	00	11
01	00	11	10
10	00	01	00
11	00	01	00
00	01	01	10
01	01	01	10
10	01	10	11
11	01	10	11
00	10	11	00
01	10	10	11
10	10	11	00
11	10	10	11
00	11	11	01
01	11	11	01
10	11	00	01
11	11	00	01

- -0,5 um ou mais erros no estado seguinte
- -0,5 um ou mais erros numa das saídas
- -0,5 um ou mais erros na codificação dos estados
- -0,5 um ou mais erros na entrada de dados ou no estado atual
- +0,5 tem uma tabela com as colunas corretas, e número de linhas correto

Poderiam ter utilizado o x na notação da tabela, para diminuir o número de linhas (ficaria igual o número de arcos no diagrama de estados).

Pergunta 2. - 2 valores -- Exame (1 valor P-fólio)

- 0,5 valores por cada mapa correto (4 variáveis no total do exame, e 2 no p-fólio)
- 0,5 no caso de erro na conversão para os mapas de Karnaugh, mas correta simplificação (p-fólio)

Houve estudantes no p-fólio a realizarem a simplificação das variáveis de saída.

Pergunta 3. - 1 valor -- Exame

- 1 Esquema corretamente desenhado, com base na alínea anterior.

Grupo IV

Pergunta 1.a) - 0,5 valores -- Exame

AND R1, M[R2]

Pergunta 1.b) - 0,5 valores -- Exame

JMP.NZ label

Pergunta 1.c) - 0,5 valores -- Exame

POP M[R1]

Pergunta 1.d) - 0,5 valores -- Exame

ROR R1, 1

Pergunta 1/2. - 3 valores

Avaliado apenas a parte da função solicitada

```
       MOV R5, R1         ; prod - R5
    MOV R6, 1  ; soma - R6
    MOV R4, 1  ; inicialmente não há erro
For: DEC R2     ; como k não é utilizado na expressão, fica como sendo a variável
iteradora
    BR.N Fim   ; no caso de k=1, nesta altura seria 0 e tem ainda de fazer uma
iteração, pelo que apenas quando é negativo é que termina
      ADD R6, R5 ; soma = soma + prod
   BR.C Erro  ; caso exista carry, significa que não há espaço para armazenar
        MOV R7, R1 ; cópia de p, para efetuar o produto
   MUL R7, R5 ; prod fica com o produto, e R7 deve ser zero
    TEST R7, R0        ; ver se R7 é de facto zero
    BR.NZ Erro ; no caso de não ser há erro
   BR For     ; ciclo terminado, passa para o próximo
Fim:    MOV R2, 1  ; R2 deixa de ser necessário para a variável iteradora
 SUB R2, R1 ; R2 fica com 1-p
      MUL R2, R6 ; soma * (1 - p)
       TEST R2, R7        ; ver se R2 é zero
     BR.NZ Erro ; não é e assim há erro
        MOV R3, R6 ; retornar o valor em R3
       RET        ; retornar
Erro:        MOV R4, R0 ; código de erro
       RET        ; retornar
```

- -1 valor no caso de não detectar o excesso em todas as operações

```
     MOV R5, R1         ; prod - R5
    MOV R6, 1  ; soma - R6
For:        DEC R2     ; como k não é utilizado na expressão, fica como sendo a
variável iteradora
    BR.N Fim   ; no caso de k=1, nesta altura seria 0 e tem ainda de fazer uma
iteração, pelo que apenas quando é negativo é que termina
      ADD R6, R5 ; soma = soma + prod
   MOV R7, R1 ; cópia de p, para efetuar o produto
   MUL R7, R5 ; prod fica com o produto, e R7 deve ser zero
    BR For     ; ciclo terminado, passa para o próximo
Fim:    MOV R3, 1  ; R3 deve ficar com o valor final
      SUB R3, R1 ; R3 fica com 1-p
      MUL R6, R3 ; R3 = soma * (1 - p), R6 deve ser zero
        RET        ; retornar, já que em R3 tem o valor final
```

- 0 - teste do excesso na soma, mas realizado em ambos os produtos, não tem penalização
- -0,5 no caso de fazer-se os produtos sem saber que ambos os operandos são alterados. O registo da esquerda (o primeiro), fica com a parte alta do produto, no caso de não ultrapassar os 16 bits, fica com zero, e não mantém o valor anterior.
- -0,5 no caso de pequenos erros sintaticos/desconhecimento do Assembly do P3, como troca de operandos, ou produto com o mesmo registo
- 1,5 valores para código globalmente bom e estruturado (as penalizações não baixam deste valor), contendo clara correspondência das instruções de assembly com o código pedido
- 1 valor para código com erros funcionais que não estejam relacionados com sintaxe do Assembly. Utilização de registos não inicializados (que não os argumentos), não separação de prod/p, e outros erros funciuonais, mas mantendo relação das instruções com o código solicitado.
- 0,5 por código com instruções sintaticamente corretas, embora incompleto ou sem correspondência com o código solicitado
- 0,5 por código com correspondência razoável com o código pedido, mas com erros sintáticos graves, como utilizar duas constantes, não aceder a posições de memória com M[]
- 0 referência M() em vez de M[]
- Indiferente terem utilizado a flag O ou C após o ADD (quem considerou este teste). Como existe uma operação de subtração, julgo que seria até mais correto utilizar a flag O, dado que é ativado desde que o resultado não possa ser representado em complemento para 2. Em todo o caso não houve de qualquer forma penalização para a omissão do teste nesta operação.
- -0,5 para teste com a flag O após a operação MUL. Ficam ambas as zero já que o resultado é colocado em ambos os operandos, nunca há overflow.