NOME: N.º: v1

Sistemas de Computação ::: LEI ::: U Minho ::: 2021/22 ::: 2022.06.14 ::: EXAME v1

Notas: Coloca o teu nome e número nas quatro páginas que compõem este teste.
Para cada pergunta, apresenta a justificação da solução, incluindo o raciocínio ou os cálculos que efetuares. Podes dar como resposta um valor numérico não simplificado (exemplo 15^{33}+73/18). Não são permitidas máquinas de calcular, computadores, telemóveis, tablets, etc.
Os testes são de resolução individual. Qualquer tentativa de fraude académica pode implicar a abertura de um processo disciplinar. Ao realizares este teste, estás a aceitar esta possível implicação.

[3 valores] A. O clube CrazyAboutSeven regista o número dos sócios em base 7, seguindo uma ordem cronológica. Quanto mais antigo é um sócio, menor é o seu número.
(A1) Após a renumeração de sócios, que implicou eliminar sócios inativos (e.g., falecidos, expulsos) ficou a saber-se que o sócio mais antigo tem o número 1, o 2.º mais antigo o número 2 e, por aí adiante, até ao sócio mais recente que tem o número 356_7. **Quantos sócios tem atualmente este clube**? [1 val.]
(A2) O novo presidente do clube, o sr. Twice, decidiu representar o seu número de sócio, 145_7, em binário (sinal e amplitude). **Qual é o número mínimo de bits que são precisos para fazer essa representação**? [1 val.]
(A3) **A que cadeia de bits corresponde o número -77_{10} na representação referida em A2**? [1 val.]

[2 valores] B. Considera que um ficheiro de som digital ocupa 9 MB, tem duração de 2m30s, e foi obtido com uma taxa de amostragem de 30 kHz.
(B1) **Qual é a resolução em bits do sinal capturado**? [1 val.]
(B2) **Qual o tamanho em KiB do ficheiro** que armazena os primeiros 2m08s do sinal que foram reconstruídos com uma taxa de 32 kHz e com resolução de 8 bits? [1 val.]

[3 valores] C. Considera uma versão reduzida da norma IEEE 754 com 12 bits: 5 bits para o expoente em excesso de 15, 6 bits para a mantissa e 1 bit para o sinal.
(C1) **Indica qual é o padrão binário correspondente ao valor $81x64_{10}$**. [1 val.]
(C2) **Qual é o valor (em decimal) do número representado pelo padrão hexadecimal $BE0_{16}$**? [1 val.]
(C3) **Qual é o valor (em decimal) do número representado pelo padrão binário $1000\ 0011\ 0110_2$**? [1 val.]

[3.5 valores] D. Um processador tem uma cache de mapeamento direto, com 64 linhas, cada uma com 8 palavras. A memória tem 2^{20} palavras. As palavras são endereçadas ao byte e ocupam 4 bytes.
(D1) **Quantos blocos existem na memória**? [0.75 val.]
(D2) **Desenha o formato dos endereços da memória principal** (incluindo os campos t, s, o) que permite mapeá-los para a cache. [0.75 val.]
(D3) **Qual é, em <u>bits</u>, o tamanho da cache**, considerando a existência do *valid bit*? [1 val.]
(D4) **Indica dois endereços de memória**, que pertençam a blocos na memória diferentes, que sejam mapeados para a linha da cache 1 ($00..01_2$). [1 val.]

[1.5 valores] E. Os compiladores são geralmente muito prudentes no que respeita às optimizações que aplicam para gerar código máquina, sendo especialmente sensíveis aos designados bloqueadores de optimização. **Dá dois exemplos, ao nível do código em C, de situações que representam bloqueios de optimização**.

NOME: N.º: v1

[2 valores] F. Considera a função `contaImpares` codificada em C e o código assembly gerado pelo gcc (opção de optimização -O0).
(F1) **Desenha o activation record desta função**. [1 val.]
(F2) **Relaciona as instruções 3, 4 e 5 (no código C) com as respetivas instruções do código assembly**. Faz o relacionamento no código assembly, indicando junto das linhas respetivas (8-35) que partes dessas instruções C são abrangidas. [1 val.]

```
1: int contaImpares(int n, int *val) {
2:   int i, total=0;
3:   for (i=0; i<n; i++)
4:     if (val[i]%2==1)
5:       total ++;
6:   return total;
7: }
```

```
# === gcc -O0 -S ... ===
 8: contaImpares:
 9:       pushl   %ebp
10:       movl    %esp, %ebp
11:       subl    $16, %esp
12:       movl    $0, -8(%ebp)
13:       movl    $0, -4(%ebp)
14:       jmp     .L2
15: .L4:  movl    -4(%ebp), %eax
16:       leal    0(,%eax,4), %edx
17:       movl    12(%ebp), %eax
18:       addl    %edx, %eax
19:       movl    (%eax), %edx
20:       movl    %edx, %eax
21:       sarl    $31, %eax
22:       shrl    $31, %eax
23:       addl    %eax, %edx
24:       andl    $1, %edx
25:       subl    %eax, %edx
26:       movl    %edx, %eax
27:       cmpl    $1, %eax
28:       jne     .L3
29:       addl    $1, -8(%ebp)
30: .L3:  addl    $1, -4(%ebp)
31: .L2:  movl    -4(%ebp), %eax
32:       cmpl    8(%ebp), %eax
33:       jl      .L4
34:       movl    -8(%ebp), %eax
35:       leave
36:       ret
```

NOME: N.º: v1

[2 valores] G. Considera que parte da memória tem os valores mostrados na figura ao lado e ainda as seguintes instruções em *assembly*:

```
movl 0x77AA0073(,%edi,4), %esi
addl $33, %esi
```

endereço	conteúdo
0x77AA0082	0x83
0x77AA0083	0x42
0x77AA0084	0x00
0x77AA0085	0xAA
0x77AA0086	0x77
0x77AA0087	0xAB
0x77AA0088	0x01
0x77AA0089	0x52
0x77AA008A	0x83
0x77AA008B	0x15

(G1) Se, no final da execução da instrução `movl`, o registo `%esi` tem o valor `0x77AA0042`, obtido na parte da memória mostrada, **qual é o conteúdo do registo `%edi`**? Apresenta o teu raciocínio. [1 val.]

(G2) **Qual é o valor armazenado no registo `%esi`**, após a execução da instrução `addl`. Apresenta os cálculos que efetuaste. [1 val.]

[3 valores] H. Considera duas funções C mutuamente recursivas e o respetivo código máquina IA32.

```
int par(unsigned int n) {
  if (n==0)
    return 1;
  return impar(n-1);
}

int impar(unsigned int n) {
  if (n==0)
    return 0;
  return par(n-1);
}
```

Endereço (hexadecimal)	**conteúdo**
FFFF0FC0	
FFFF0FC4	
FFFF0FC8	
FFFF0FCC	
FFFF0FD0	
FFFF0FD4	
FFFF0FD8	
FFFF0FDC	
FFFF0FE0	
FFFF0FE4	
FFFF0FE8	
FFFF0FEC	
FFFF0FF0	
FFFF0FF4	
FFFF0FF8	
FFFF0FFC	
FFFF1000	
FFFF1004	

```
<par>
0x080483e4: push %ebp
0x080483e5: movl %esp,%ebp
0x080483e7: subl $8,%esp
0x080483ea: cmpl $0,0x8(%ebp)
0x080483ee: jne 0x80483f9 <par+21>
0x080483f0: movl $1,-4(%ebp)
0x080483f7: jmp 0x804840a <par+38>
0x080483f9: movl 8(%ebp),%eax
0x080483fc: subl $1,%eax
0x080483ff: movl %eax,(%esp)
0x08048402: call 0x804840f <impar>
0x08048407: movl %eax,-4(%ebp)
0x0804840a: movl -4(%ebp),%eax
0x0804840d: leave
0x0804840e: ret
<impar>
0x0804840f: push %ebp
0x08048410: movl %esp,%ebp
0x08048412: subl $8,%esp
0x08048415: cmpl $0,0x8(%ebp)
0x08048419: jne 0x8048424 <impar+21>
0x0804841b: movl $0,-4(%ebp)
0x08048422: jmp 0x8048435 <impar+38>
0x08048424: movl 8(%ebp),%eax
0x08048427: subl $1,%eax
0x0804842a: movl %eax,(%esp)
0x0804842d: call 0x80483e4 <par>
0x08048432: movl %eax,-4(%ebp)
0x08048435: movl -4(%ebp),%eax
0x08048438: leave
0x08048439: ret
```

(H1) Assume que foi feita uma chamada a `par(2)`, que vai desencadear as seguintes invocações: `par(2)`, `impar(1)` e `par(0)`. Para a chamada `par(2)`, os valores de `%esp` e `%ebp`, antes da invocação, são `0xFFFF1000` e `0xFFFF1022` e o endereço de regresso é `0x080483C9`. **Preenche o conteúdo completo da pilha** para a execução dessa sequência de chamadas, i.e., até ao instante que antecede a execução da instrução `ret` no contexto de `par(0)`. [2 val.]

(H2) **Quais são os valores de `%esp` e `%ebp`**, imediatamente antes da execução da instrução `ret` no contexto `impar(1)`? [1 val.] `%esp` = 0x_____ `%ebp` = 0x_____

NOME: N.º: v2

Sistemas de Computação ::: LEI ::: U Minho ::: 2021/22 ::: 2022.06.14 ::: EXAME v2

Notas: Coloca o teu nome e número nas quatro páginas que compõem este teste.
Para cada pergunta, apresenta a justificação da solução, incluindo o raciocínio ou os cálculos que efetuares. Podes dar como resposta um valor numérico não simplificado (exemplo 15^{33}+73/18). Não são permitidas máquinas de calcular, computadores, telemóveis, tablets, etc.
Os testes são de resolução individual. Qualquer tentativa de fraude académica pode implicar a abertura de um processo disciplinar. Ao realizares este teste, estás a aceitar esta possível implicação.

[3 valores] A. O clube CrazyAboutSeven regista o número dos sócios em base 7, seguindo uma ordem cronológica. Quanto mais antigo é um sócio, menor é o seu número.
(A1) Após a renumeração de sócios, que implicou eliminar sócios inativos (e.g., falecidos, expulsos) ficou a saber-se que o sócio mais antigo tem o número 1, o 2.º mais antigo o número 2 e, por aí adiante, até ao sócio mais recente que tem o número 355_7. **Quantos sócios tem atualmente este clube**? [1 val.]
(A2) O novo presidente do clube, o sr. Twice, decidiu representar o seu número de sócio, 146_7, em binário (sinal e amplitude). **Qual é o número mínimo de bits que são precisos para fazer essa representação**? [1 val.]
(A3) **A que cadeia de bits corresponde o número -78_{10} na representação referida em A2**? [1 val.]

[2 valores] B. Considera que um ficheiro de som digital ocupa 9 MB, tem duração de 2m30s, e foi obtido com uma taxa de amostragem de 30 kHz.
(B1) **Qual é a resolução em bits do sinal capturado**? [1 val.]
(B2) **Qual o tamanho em KiB do ficheiro** que armazena os primeiros 2m08s do sinal que foram reconstruídos com uma taxa de 32 kHz e com resolução de 12 bits? [1 val.]

[3 valores] C. Considera uma versão reduzida da norma IEEE 754 com 12 bits: 5 bits para o expoente em excesso de 15, 6 bits para a mantissa e 1 bit para o sinal.
(C1) **Indica qual é o padrão binário correspondente ao valor $82x64_{10}$**. [1 val.]
(C2) **Qual é o valor (em decimal) do número representado pelo padrão hexadecimal $BE1_{16}$**? [1 val.]
(C3) **Qual é o valor (em decimal) do número representado pelo padrão binário $1000\ 0011\ 0110_2$**? [1 val.]

[3.5 valores] D. Um processador tem uma cache de mapeamento direto, com 64 linhas, cada uma com 8 palavras. A memória tem 2^{20} palavras. As palavras são endereçadas ao byte e ocupam 4 bytes.
(D1) **Quantos blocos existem na memória**? [0.75 val.]
(D2) **Desenha o formato dos endereços da memória principal** (incluindo os campos t, s, o) que permite mapeá-los para a cache. [0.75 val.]
(D3) **Qual é, em <u>bits</u>, o tamanho da cache**, considerando a existência do *valid bit*? [1 val.]
(D4) **Indica dois endereços de memória**, que pertençam a blocos na memória diferentes, que sejam mapeados para a linha da cache 3 ($00..011_2$). [1 val.]

[1.5 valores] E. Os compiladores são geralmente muito prudentes no que respeita às optimizações que aplicam para gerar código máquina, sendo especialmente sensíveis aos designados bloqueadores de optimização. **Dá dois exemplos, ao nível do código em C, de situações que representam bloqueios de optimização**.

NOME: N.º: v2

[2 valores] F. Considera a função `contaImpares` codificada em C e o código assembly gerado pelo gcc (opção de optimização -O0).
(F1) **Desenha o activation record desta função**. [1 val.]
(F2) **Relaciona as instruções (c), (d) e (e) (no código C) com as respetivas instruções do código assembly**. Faz o relacionamento no código assembly, indicando junto das linhas respetivas (1-29) que partes dessas instruções C são abrangidas. [1 val.]

```
a: int contaImpares(int n, int *val) {
b:   int i, total=0;
c:   for (i=0; i<n; i++)
d:     if (val[i]%2==1)
e:       total ++;
f:   return total;
g: }
```

```
# === gcc -O0 -S ... ===
 1: contaImpares:
 2:       pushl   %ebp
 3:       movl    %esp, %ebp
 4:       subl    $16, %esp
 5:       movl    $0, -8(%ebp)
 6:       movl    $0, -4(%ebp)
 7:       jmp     .L2
 8: .L4:  movl    -4(%ebp), %eax
 9:       leal    0(,%eax,4), %edx
10:       movl    12(%ebp), %eax
11:       addl    %edx, %eax
12:       movl    (%eax), %edx
13:       movl    %edx, %eax
14:       sarl    $31, %eax
15:       shrl    $31, %eax
16:       addl    %eax, %edx
17:       andl    $1, %edx
18:       subl    %eax, %edx
19:       movl    %edx, %eax
20:       cmpl    $1, %eax
21:       jne     .L3
22:       addl    $1, -8(%ebp)
23: .L3:  addl    $1, -4(%ebp)
24: .L2:  movl    -4(%ebp), %eax
25:       cmpl    8(%ebp), %eax
26:       jl      .L4
27:       movl    -8(%ebp), %eax
28:       leave
29:       ret
```

NOME: N.º: v2

[2 valores] G. Considera que parte da memória tem os valores mostrados na figura ao lado e ainda as seguintes instruções em *assembly*:

```
movl 0x77AA0073(,%edi,4), %esi
addl $35, %esi
```

endereço	conteúdo
0x77AA0082	0x83
0x77AA0083	0x42
0x77AA0084	0x00
0x77AA0085	0xAA
0x77AA0086	0x77
0x77AA0087	0xAB
0x77AA0088	0x01
0x77AA0089	0x52
0x77AA008A	0x83
0x77AA008B	0x15

(G1) Se, no final da execução da instrução `movl`, o registo `%esi` tem o valor `0x77AA0042`, obtido na parte da memória mostrada, **qual é o conteúdo do registo `%edi`**? Apresenta o teu raciocínio. [1 val.]

(G2) **Qual é o valor armazenado no registo `%esi`**, após a execução da instrução `addl`. Apresenta os cálculos que efetuaste. [1 val.]

[3 valores] H. Considera duas funções C mutuamente recursivas e o respetivo código máquina IA32.

```
int par(unsigned int n) {
  if (n==0)
    return 1;
  return impar(n-1);
}

int impar(unsigned int n) {
  if (n==0)
    return 0;
  return par(n-1);
}
```

```
<par>
0x080483e4: push %ebp
0x080483e5: movl %esp,%ebp
0x080483e7: subl $8,%esp
0x080483ea: cmpl $0,0x8(%ebp)
0x080483ee: jne 0x80483f9 <par+21>
0x080483f0: movl $1,-4(%ebp)
0x080483f7: jmp 0x804840a <par+38>
0x080483f9: movl 8(%ebp),%eax
0x080483fc: subl $1,%eax
0x080483ff: movl %eax,(%esp)
0x08048402: call 0x804840f <impar>
0x08048407: movl %eax,-4(%ebp)
0x0804840a: movl -4(%ebp),%eax
0x0804840d: leave
0x0804840e: ret
<impar>
0x0804840f: push %ebp
0x08048410: movl %esp,%ebp
0x08048412: subl $8,%esp
0x08048415: cmpl $0,0x8(%ebp)
0x08048419: jne 0x8048424 <impar+21>
0x0804841b: movl $0,-4(%ebp)
0x08048422: jmp 0x8048435 <impar+38>
0x08048424: movl 8(%ebp),%eax
0x08048427: subl $1,%eax
0x0804842a: movl %eax,(%esp)
0x0804842d: call 0x80483e4 <par>
0x08048432: movl %eax,-4(%ebp)
0x08048435: movl -4(%ebp),%eax
0x08048438: leave
0x08048439: ret
```

Endereço (hexadecimal)	**conteúdo**
FFFF0FC0	
FFFF0FC4	
FFFF0FC8	
FFFF0FCC	
FFFF0FD0	
FFFF0FD4	
FFFF0FD8	
FFFF0FDC	
FFFF0FE0	
FFFF0FE4	
FFFF0FE8	
FFFF0FEC	
FFFF0FF0	
FFFF0FF4	
FFFF0FF8	
FFFF0FFC	
FFFF1000	
FFFF1004	

(H1) Assume que foi feita uma chamada a `par(2)`, que vai desencadear as seguintes invocações: `par(2)`, `impar(1)` e `par(0)`. Para a chamada `par(2)`, os valores de `%esp` e `%ebp`, antes da invocação, são `0xFFFF1000` e `0xFFFF1022` e o endereço de regresso é `0x080483C9`. **Preenche o conteúdo completo da pilha** para a execução dessa sequência de chamadas, i.e., até ao instante que antecede a execução da instrução `ret` no contexto de `par(0)`. [2 val.]

(H2) **Quais são os valores de `%esp` e `%ebp`**, imediatamente antes da execução da instrução `ret` no contexto `impar(1)`? [1 val.] `%esp` = 0x_____ `%ebp` = 0x_____