Sistemas Embebidos I: Colecção de Exercícios

Tiago M. Dias

Dezembro de 2009

Instituto Superior de Engenharia de Lisboa
Departamento de Engenharia de Electrónica e de Telecomunicações e de Computadores
Secção de Arquitecturas e Sistemas Operativos

Edição 1.2

Prefácio

Esta edição do texto "Sistemas Embebidos I: Colecção de Exercícios" destina-se a apoiar o ensino da disciplina de *Sistemas Embebidos I*, leccionada aos cursos de Licenciatura em Engenharia de Electrónica e Telecomunicações e de Computadores, Licenciatura em Engenharia Informática e de Computadores, Mestrado em Engenharia de Electrónica e Telecomunicações e Mestrado em Engenharia Informática e de Computadores do Instituto Superior de Engenharia de Lisboa (ISEL).

A colecção inclui, essencialmente, exercícios não-resolvidos das matérias respeitantes a cada um dos capítulos em que se desdobra o programa da disciplina de Sistemas Embebidos I. Os problemas apresentados resultam de uma compilação de exercícios já apresentados anteriormente em enunciados de testes da disciplina de *Sistemas Embebidos I*, bem como da disciplina de *Microprocessaores I* do antigo curso de Bacharelato em Engenharia Informática e de Computadores do ISEL. A apresentação dos exercícios é feita por grupo temático e, dentro de cada grupo, por ordem cronológica.

Futuramente, e sempre que oportuno, a colecção poderá vir a ser actualizada e/ou ampliada com a inclusão de novos problemas.

Tiago M. Dias
Janeiro de 2008

Versão 1.2, editada por Tiago M. Dias em Dezembro de 2009
Versão 1.1, editada por Tiago M. Dias em Setembro de 2008
Versão 1.0, editada por Tiago M. Dias em Janeiro de 2008

Conteúdo

Capítulo 1

Arquitectura ARM

1.1 Caracterização da arquitectura ARM

1. [*2003/04v, 1º Teste*]
   Relativamente ao código de baixo nível para processamento de uma interrupção, escrito em assembly, que foi apresentado nas aulas e utilizado nos trabalhos,

   a) Descreva o que faz este código antes e depois de executar a função de processamento de alto nível.

   b) Por que razão é necessário este código e não se associa directamente, à interrupção, a função de alto nível?

2. [*2004/05i, 1º Teste*]
   Quais as vantagens que decorrem de as *words* em memória serem obrigatoriamente localizadas em endereços múltiplos de 4?

3. [*2004/05i, 2º Teste*]
   Porque é que a arquitectura ARM tem limitações na codificação de valores imediatos?

4. [*2004/05i, 2º Teste*]
   Defina a organização de memória *little-endian*.

5. [*2004/05v, 1º Teste*]
   Sabendo que, na arquitectura ARM, os acessos a *stack* são realizados por instruções de `load` e `store` usando um registo como *stack pointer*, diga, justificando, se há alguma razão para se ter escolhido o registo `R13` para este fim ou se poderia ter sido outro.

6. [*2004/05v, 2º Teste*]
   No prólogo APCS, o registo `R12` é usado para armazenar temporariamente o valor anterior de `SP`.

   Poderia ser outro registo qualquer? Justifique.

7. [*2005/06i, 2º Teste*]
   Relativamente ao processamento de interrupções na arquitectura ARM,

   a) Indique as alterações introduzidas ao estado do processador pela execução da instrução `ldmfd sp!, pc^` no final de uma rotina de serviço de interrupção.

   b) Justifique a necessidade de uma instrução atómica, tal como a indicada na alínea anterior, para finalizar a execução de uma rotina de serviço de interrupção.

8. [*2005/06v, 2º Teste*]
   Um sistema baseado num processador de arquitectura ARM utiliza um *stack* auxiliar com filosofia *empty descending*, utilizando o registo `R10` como *stack pointer*. Apresente as instruções assembly do ARM equivalentes às operações `PUSH R0` e `POP R0` neste *stack*.

9. [*2007/08i, 1º Teste*]
   Relativamente à arquitectura ARM,

a) Justifique a necessidade de duplicação dos registos `R13`, `R14` e `SPSR` nos modos de excepção.

b) Apresente a motivação que levou a que os registos `R8` a `R12` fossem também duplicados no modo de excepção *Fast Interrupt* (`FIQ`).

c) Indique as principais vantagens da escolha do endereço de memória `0x1C` para a localização do vector de excepção `FIQ`.

1.2 Elaboração de ficheiros executáveis

1. [*2004/05i, 2º Teste*]
   Na realização de programas são geradas as secções `.text`, `.data`, `.rodata` e `.bss`. Descreva o fim a que se destina cada uma delas. Dê exemplos de código em C que lhes dão origem.

2. [*2005/06v, 1º Teste*]
   O quadro seguinte apresenta o resultado de uma execução do comando `make`. Sabendo que no ficheiro `main.cpp` está definida a função com a assinatura `void isr_c()`, explique a causa do erro e indique, justificando, de que forma(s) este pode ser corrigido.

```
$ make
arm-elf-as --gstabs -o cstart.o cstart.s
arm-elf-gcc -c -g -o crt_init.o crt_init.cpp
arm-elf-as --gstabs -o isr.o isr.s
arm-elf-gcc -c -g -o main.o main.cpp
arm-elf-ld cstart.o crt_init.o isr.o main.o -o main.x -T ldscript -Map
main.map
isr.o(.text+0x8):isr.s:8: undefined reference to `isr_c'
make: *** [main.x] Error 1
```

3. [*2005/06v, 2º Teste*]
   Os ficheiros `types.h` e `at91m55800.h`, utilizados nos trabalhos práticos, contêm as seguintes definições:

```
typedef unsigned long U32;
volatile U32 * const PIOB = (U32 *) 0xffff0000;
```

   Porque razão é utilizado o qualificador `volatile` na definição de `PIOB`? E o qualificador `const`?

4. [*2005/06v, 3º Teste*]
   O quadro seguinte apresenta o resultado de uma execução do comando `make`. Sabendo que no ficheiro `modasm.s` está definida uma *label* `a` sobre uma *word* na secção `.data`, explique a causa do erro e indique, justificando, de que forma(s) este pode ser corrigido.

```
$ make
arm-elf-as --gstabs -o cstart.o cstart.s
arm-elf-as --gstabs -o modasm.o modasm.s
arm-elf-gcc -c -g -o modc.o modc.c
arm-elf-ld cstart.o modasm.o modc.o -o app.x -T ldscript
modc.o(.text+0x20): In function `main':
modc.c:7: undefined reference to `a'
make: *** [modc.x] Error 1
```

5. [*2007/08i, 1º Teste*]
   O quadro seguinte apresenta o conteúdo do ficheiro `Makefile` associado à geração da aplicação `programa`.

```
programa: cstart.o crt_init.o main.o ldscript makefile
        arm-elf-ld cstart.o crt_init.o main.o -o programa

cstart.o: cstart.s makefile
```

```
        arm-elf-as --gstabs cstart.s -o cstart.o

crt_init.o: crt_init.cpp makefile
        arm-elf-gcc -c -g -o crt_init.o crt_init.cpp

main.o: main.c makefile
        arm-elf-gcc -c -g main.c -o main.o
```

a) Descreva, sucintamente, a estrutura básica de programação associada a um ficheiro `Makefile`.

b) Relativamente ao método de geração de ficheiros executáveis, discuta as principais vantagens da utilização da aplicação `make` e dos ficheiros de *script* `Makefile` por oposição à utilização de ficheiros *batch*.

6. [*2007/08i, 2º Teste*]
O quadro abaixo apresenta o resultado de uma execução do comando `arm-elf-as` sobre o ficheiro `x.s`, escrito em *assembly* da arquitectura ARM.

```
$ arm-elf-as x.s -o x.o
x.s: Assembler messages:
x.s:5: Error: invalid constant (565623) after fixup
```

Considerando que o conteúdo do ficheiro `x.s` é o seguinte:

```
        .text
        .global _start
_start:
        mov r1, #1
        mov r0, #0x565623
        add r0, r0, r1
```

a) Explique, justificando, a causa do erro.

b) Indique, justificando, de que forma(s) o erro pode ser corrigido.

7. [*2007/08i, 3º Teste*]
Considere o seguinte troço de código em linguagem C:

```
typedef struct { long ID; char name[20]; short number; char message[100]; } student_t;

extern student_t class[10];

char buffer[50];
char message[] = ". Boa sorte para o teste!";

int alunos=10;

int main(void)
{
        int i, size=0;
        for(i=0; i<alunos; ++i)
        {
                strcpy(buffer, "Ola ");
                strcat(buffer, class[i].name);
                strcat(buffer, message);
                strcpy(class[i].message, buffer);
        }

        return 0;
}
```

a) Indique, justificando, as secções em que serão localizadas as variáveis `buffer`, `message`, `alunos`, `size` e `i`.

b) Represente a organização em memória de uma variável do tipo de dados `student_t`.

8. [*2008/09i, 1º Teste*]
Considere que o seguinte troço de programa é compilado com o compilador GNU invocado com o comando: `arm-elf-gcc -c main.c -o main.o`.

```
#define X 5
#define Y 1
char string[33];

int main() {
        static int x = X, y = Y;
        struct tm tm;
        rtc_read(&tm);
        strftime(string, sizeof(string), "%A %d %b %Y %H:%M:%S", &tm);
        lcd_clear();
        lcd_write_string(x, y, string);
        return 0;
}
```

a) Qual a dimensão das secções resultantes `.data`, `.rodata` e `.bss`?

b) Se se pretender executar o programa de memória ROM quais as secções que deverão ser gravadas na ROM? Justifique.

9. [*2008/09i, 1º Teste*]
Porque é que na gravação de programas em ROM utilizamos o formato binário e não o formato ELF? Seria possível executar um programa gravado em formato ELF? Justifique.

10. [*2008/09i, 2º Teste*]
Porque é que na preparação do programa para gravação em ROM é necessário alterar o módulo de arranque (`cstart.s` para `cstart_rom.s`) e o *script* de localização (`ldscript` para `ldscript_rom`)?

1.3 Assembly ARM

1. [*2003/04v, 2º Teste*]
Escreva, em assembly da arquitectura ARM, o código da função

```
unsigned getFiqRegisters(U32 buffer[]);
```

que preenche os elementos do *array* `buffer` com os valores dos registos específicos do modo de execução FIQ (`R8_fiq` a `R14_fiq` e `SPSR_fiq`).

2. [*2004/05i, 1º Teste*]
Escreva, em assembly da arquitectura ARM, a função `toBCD` que converte um valor binário para BCD representado a 32-bit.

```
int toBCD(int);
```

3. [*2004/05i, 1º Teste*]
Escreva, em assembly da arquitectura ARM, a função `toBCD` que converte um valor binário para BCD representado a 32-bit.

```
int toBCD(int);
```

4. [*2004/05i, 1º Teste*]
Escreva, em assembly da arquitectura ARM, a função `seqlen` que determina a dimensão da sequência de inteiros apontada por `seq`. O fim da sequência é definido pela ocorrência do valor `MAXINT`.

```
int seqlen(int * seq);
```

5. [*2004/05i, 2º Teste*]
   Escreva, em assembly da arquitectura ARM, o código da função `strSubst` que modifica a *string* `str`, substituindo pelo carácter espaço qualquer carácter que pertença à *string* `match`.

   ```
   int strSubst(char * str, char * match);
   ```

6. [*2005/06i, 1º Teste*]
   Escreva, em assembly da arquitectura ARM, o código da função `sum2`, que retorna na posição apontada por `acc1` o resultado da soma do conteúdo de `acc1` com `val1` e na posição apontada por `acc2` o resultado da soma do conteúdo de `acc2` com `val2`.

   ```
   void sum2(int * acc1, int * acc2, int val1, int val2);
   ```

7. [*2005/06v, 2º Teste*]
   Considere as seguintes definições:

   ```
   typedef struct {
           char valid;
           char id[2];
           unsigned int dim;
           void * data;
   } Block;

   unsigned int BlockArrayCompact(Block * blockArray, unsigned int nElems);
   ```

   Escreva, em assembly da arquitectura ARM, o código da função `BlockArrayCompact` que recebe um *array* de `Blocks` com `nElems` posições e remove os elementos cujo campo `valid` tenha o valor 0. A remoção é feita movendo os elementos válidos para posições contíguas a partir do início do *array*. A função retorna o número de objectos válidos encontrados.

8. [*2006/07i, 1º Teste*]
   Programe em assembly ARM a função `get_field` que retorna o valor inteiro extraído de uma gama de bits do parâmetro `word`, a partir da posição `offset`, e com a dimensão `size`.

   ```
   int get_field(int word, int size, int offset);
   ```

   Exemplo: `word`=0x12345678, `offset`=10, `size`=5, retorna 0x15.

9. [*2006/07i, 1º Teste*]
   Programe em assembly ARM, a função `csum` que efectua a soma da sequência de bytes definida pelos parâmetros `data` e `size`.

   ```
   char csum(char * data, int size);
   ```

10. [*2006/07i, 2º Teste*]
    Programe em assembly ARM a função `set_field` que retorna o valor do parâmetro `word` modificado pela afectação, com o valor `value`, da gama de bits definida a partir da posição `offset`, e com a dimensão `size`.

    ```
    int set_field(int word, int size, int offset, int value);
    ```

    Exemplo: `word`= 0x12345678, `size`= 5, `offset`= 10, `value`= 0xA, retorna 0x12342A78

11. [*2006/07i, 2º Teste*]
    Programe em assembly ARM, de uma forma eficiente, a função `split` para separar as amostras PCM de um canal estereofónico.

    ```
    void split(int * stereo, int len, int * left, int * right);
    ```

    O parâmetro `stereo` é um ponteiro para uma sequência de valores inteiros que representam as amostras da via esquerda e da via direita, colocadas alternadamente. O parâmetro `len` indica o número de amostras de cada via. As amostras são colocadas em duas sequências separadas indicadas respectivamente pelos ponteiros `left` e `right`.

12. [*2007/08i, 1º Teste*]
Realize, em *assembly* da arquitectura ARM, a função

```
int strCapitalize(char *str);
```

que muda para maiúscula a primeira letra de cada palavra contida na *string* `str`. A *string* `str` é composta por uma ou mais palavras, podendo as palavras estar separadas por um ou mais caracteres de espaçamento ' ' (' ' corresponde ao valor `0x20` no código ASCII). A função `strCapitalize` devolve o número de palavras que foram modificadas.
Para localizar caracteres na *string* `str` deve utilizar a função de biblioteca

```
char * strchr(const char *cs, const char ch);
```

que retorna o apontador para o primeiro caractere `ch` na string `cs`, ou `NULL` se esse caractere não existir.

13. [*2007/08i, 2º Teste*]
Considere as seguintes definições:

```
typedef struct {
        long l;
        char ch;
        void (*func)(long l, char ch);
} something_t;

int CopyNSomething(something_t *src, something_t *dst, int nelem);
```

Escreva, em *assembly* da arquitectura ARM, o código da função `CopyNSomething` que copia o conteúdo dos `nelem` do *array* de estruturas `something_t` apontado por `src`, para o *array* de estruturas `something_t` apontado por `dst`. A função devolve o número de *bytes* copiados.

14. [*2007/08i, 3º Teste*]
Escreva, em *assembly* da arquitectura ARM e respeitando a norma APCS, a função

```
int ToBCD(unsigned int val);
```

que devolve o valor em BCD do parâmetro `val`.

Nota: Utilize as funções `Div` e `Mod` realizadas no exercício anterior para as operações de divisão.

15. [*2007/08i, 3º Teste*]
Escreva, em *assembly* da arquitectura ARM e respeitando a norma APCS, as funções

```
int Div(unsigned int val, unsigned int d);
int Mod(unsigned int val, unsigned int d);
```

A função `Div` devolve o resultado da divisão inteira do valor passado no parâmetro `val` pelo valor passado no parâmetro `d`. A função `Mod` devolve o resto da divisão inteira do valor passado no parâmetro `val` pelo valor passado no parâmetro `d`. Caso o parâmetro `d` seja zero, as funções devem retornar -1.

16. [*2008/09i, 1º Teste*]
Realize, em linguagem *assembly* ARM, a função `exchange`, em conformidade com a norma APCS. Esta função troca os valores apontados pelos ponteiros `pi` e `pj`.

```
void exchange(int * pi, int * pj);
```

17. [*2008/09i, 1º Teste*]
Traduza a função `bubble_sort` para linguagem *assembly* ARM respeitando a norma APCS. Esta função ordena um *array* de valores inteiros por ordem crescente.

```
void bubble_sort(int * base, unsigned int size) {
        int i, j, * p;
        for (i = 0; i < size - 1; ++i)
                for (j = 0, p = base; j < size - 1 - i; ++j, ++p)
                        if (*p > *(p + 1))
                                exchange(p, p + 1);
}
```

18. [*2008/09i, 2º Teste*]
Escreva, em linguagem *assembly* ARM, a função `student_compare`, em conformidade com a norma APCS. Esta função compara as notas (campo `grade`) de dois alunos, representados por `s1` e `s2`. Devolve 0 quando as notas são iguais, um valor positivo quando a nota de `s1` é melhor que a nota de `s2`, e um valor negativo no caso contrário. As notas são representadas de A a F, sendo A a classificação melhor e F a pior.

```
typedef struct _student {
        char * name; short number; char grade;
} student;

int student_compare(student * s1, student * s2);
```

19. [*2008/09i, 2º Teste*]
Traduza para linguagem *assembly* ARM, respeitando a norma APCS, a função `bsearch` pertencente à biblioteca normalizada da linguagem C. Esta função faz a pesquisa dicotómica de um elemento num *array* de elementos ordenado por ordem crescente.

```
void * bsearch(void * key, void * base, size_t nelem, size_t size,
               int (* fcmp)(void  *, void  *)) {
  while (nelem > 0) {
    size_t pivot = nelem / 2;
    char *q = (char *)base + size * pivot;
    int val = (*fcmp)(key, q);

    if (val == 0)
      return ((void *) q);
    else if (val < 0)
      nelem = pivot;
    else {
      base = q + size;
      nelem -= pivot + 1;
    }
  }
  return NULL;
}
```

1.4 Suporte para linguagens de alto nível

1. [*2003/04v, 1º Teste*]
Escreva, em assembly da arquitectura ARM, o código da função

```
int strTranslate(char * str, char * old, char * new);
```

que substitui, na *string* `str`, qualquer carácter pertencente à *string* `old` pelo carácter com a posição correspondente na *string* `new`. A função devolve o número de caracteres substituídos; Se `old` e `new` tiverem dimensões diferentes, devolve -1. Para obter as dimensões das *strings*, utilize a função de biblioteca

```
int strlen(char *);
```

2. [*2003/04v, 2º Teste*]
Escreva, em assembly da arquitectura ARM, o código da função

```
int arrayCountElem(void * base, int nel, int elsz, void * match);
```

que conta o número de elementos iguais a `match` que existem no *array* apontado por base.
Os parâmetros `nel` e `elsz` representam o número de elementos e a dimensão, em bytes, de cada elemento.
Para realizar a comparação dos elementos, utilize a função de biblioteca

```
int memcmp(void *src1, void *src2, int size);
```

esta função devolve 0 se os elementos forem iguais.

3. [*2004/05i, 1º Teste*]
Escreva, em assembly da arquitectura ARM, a função `seqlen` que determina a dimensão da sequência de inteiros apontada por `seq`. O fim da sequência é definido pela ocorrência do valor `MAXINT`.

```
int seqlen(int * seq);
```

4. [*2004/05i, 2º Teste*]
Escreva, em assembly da arquitectura ARM, o código da função `strSubst` de forma que a substituição não seja *case-sensitive*. A função `strSubst` modifica a *string* `str`, substituindo pelo carácter espaço qualquer carácter que pertença à *string* `match`.

```
int strSubst(char * str, char * match);
```

Para isso, antes de comparar os caracteres converta-os para minúsculas utilizando a função `tolower` da biblioteca.

```
int tolower(int);
```

5. [*2004/05v, 1º Teste*]
Escreva, em assembly da arquitectura ARM, o código da função

```
bool capicua(const char * str);
```

que retorna `true` (valor 1) se a *string* `str` for uma capicua.
Para determinar a dimensão da *string*, use a função de biblioteca

```
int strlen(const char * cs);
```

que devolve o número de caracteres que compõe a *string* `cs`.

6. [*2006/07i, 1º Teste*]
Escreva, em assembly da arquitectura ARM, a função

```
int copy_if(const char * orig[], char dest[][10], char * str, int n);
```

O parâmetro `orig` é um *array* de *pointers* para *string*. O parâmetro `dest` é um *array* de blocos de memória. O parâmetro `n` representa o número de elementos de `orig`.
A função compara as *strings* de `orig` com `str` e copia para `dest` as que forem diferentes, depositando-as em blocos consecutivos. Retorna o número de cópias efectuadas.
Use as funções de biblioteca com os protótipos seguintes:

```
int strcmp(const char * str1, const char * str2);
```

Compara duas *strings* e devolve 0 se são iguais.

```
char * strcpy(char * dst, const char * src);
```

Copia a *string* apontada por `src` para o endereço `dst`.

7. [*20056/06i, 1º Teste*]
Considere a seguinte definição:

```
typedef struct {
        int first;
        int second;
} pair;
```

Escreva, em assembly da arquitectura ARM, o código da função `array_sum2` que retorna o resultado do somatório dos `count` elementos que compõem o *array* `values`.

```
pair array_sum2(pair values[], unsigned int count);
```

Utilize a função `sum2` para o cálculo das somas parciais.

```
void sum2(int * acc1, int * acc2, int val1, int val2);
```

8. [*2006/07i, 1º Teste*]
Considere as seguintes definições:

```
struct ListElement {
        void * data;
        ListElement * next;
};

typedef void (*fptr)(void * data, void * context);
```

Escreva, em assembly da arquitectura ARM, o código da função `for_each` que aplica a função `action` a cada elemento da lista simplesmente ligada `list`. A função indicada em `action` recebe como primeiro parâmetro o campo `data` de um elemento da lista e `context` como segundo parâmetro. O campo `next` do último elemento da lista simplesmente ligada tem o valor 0.

```
void for_each(ListElement * list, void * context, fptr action);
```

9. [*2005/06i, 1º Teste*]

   a) Considere a seguinte definição:

```
typedef struct {
        char nome[30];
        int  numero;
        unsigned char turma;
} Aluno;
```

   Escreva, em assembly da arquitectura ARM, o código da função `verifica_turma` que retorna 1 se o campo `turma` de `a` for igual ao parâmetro `t` e 0 caso contrário.

```
int verifica_turma(Aluno *a, unsigned char t);
```

   b) Escreva, em assembly da arquitectura ARM, o código da função `remove_copy_if` que copia para `dest` os elementos do *array* [`ini`, `fin`[que não verificam a condição `pred`. Cada elemento da sequência tem dimensão `dim` bytes. `pred` recebe, como primeiro argumento, um ponteiro para um elemento da sequência e, como segundo argumento, o valor passado em `context`, retornando 1 se o primeiro argumento satisfizer a condição parametrizada pelo segundo argumento, ou 0 caso contrário. A função `remove_copy_if` devolve o número de elementos copiados.

```
int remove_copy_if(void * ini, void * fin, unsigned int dim,
    void * dest, int (*pred)(void *, void *), void * context);
```

   c) Implemente, na linguagem C ou C++, a função `RetiraTurma`, que copia todas as entradas do *array* `alunos` para `resultado`, excepto as respeitantes à `turma` indicada. O número de elementos do *array* é indicado em `numAlunos`. Utilize como predicado a função `verifica_turma`, explicando porque o pode fazer. A função `RetiraTurma` devolve o número de elementos copiados.

```
int RetiraTurma(Aluno * alunos, int numAlunos, Aluno * resultado,
                unsigned char turma);
```

10. [*2005/06v, 3º Teste*]
Considere as seguintes definições:

```
typedef struct {
        long * numero;
        Data validade;
} Acesso;
```

```
typedef struct {
        unsigned char mes;
        unsigned char ano;
} Data;
```

a) Escreva, em assembly da arquitectura ARM, o código da função `validade_acesso` que retorna 1 se os campos `mes` e `ano` de `a` registarem uma data inferior à indicada em `d` e 0 caso contrário.

```
int validade_acesso(Acesso *a, Data *d);
```

b) Implemente, na linguagem C++, a função `ActualizaAcessos` que substitui por `cancelado` todas as entradas do *array* `acessos` com campo `validade` inferior à indicada no parâmetro com o mesmo nome. O número de elementos do *array* é indicado em `numAcessos`. A função `ActualizaAcessos` devolve o número de elementos substituidos. Para a implementação, recorra à função `replace_if_not`, utilizando como predicado a função `validade_acesso`.

```
int ActualizaAcessos(Acesso * acessos, unsigned numAcessos,
                     Data * validade, Acesso * cancelado);
```

11. [*2005/06v, 3º Teste*]
Escreva, em assembly da arquitectura ARM, o código da função `replace_if_not` que substitui por `val` todos os elementos do *array* [`ini`, `fin`[que não verificam a condição `pred`. Cada elemento do *array* tem a dimensão `dim`. `pred` recebe, como primeiro argumento, um ponteiro para um elemento da sequência e, como segundo argumento, o valor passado em `context`, retornando 1 se o primeiro argumento satisfizer a condição parametrizada pelo segundo argumento, ou 0 caso contrário. A função `replace_if_not` devolve o número de elementos substituídos.

```
int replace_if_not(void * ini, void * fin, unsigned dim, const void * val,
                   void * context, int (*pred)(void *, void *));
```

12. [*2007/08i, 2º Teste*]

Relativamente à norma APCS da arquitectura ARM,

a) Identifique os registos cujo conteúdo deve ser preservado na chamada a funções e justifique a sua escolha.

b) Indique, justificando, se é possível utilizar esta norma para a escrita de funções de atendimento e de encaminhamento de interrupções.

c) Comente a afirmação: *"A norma APCS reserva apenas quatro registos para passagem de parâmetros a funções, pelo que na arquitectura ARM uma função só poderá receber, no máximo, quatro parâmetros."*

Capítulo 2

Sistema de Desenvolvimento AT91EB55

2.1 EBI

1. [*2003/04v, 1º Teste*]
   Projecte uma placa de expansão de memória para o sistema AT91EB55, formada por 4 Mbyte de RAM, utilizando um *chip* K6F3216T6M-EF55, com a capacidade de 4 Mbyte organizados em $2M \times 16 - bit$, com os seguintes sinais de interface: dados, I/O1-16; endereços, A0-20; controlo, OE#, WE#, CS1#, CS2, UB# e LB# ("# "significa *active low*). As tabelas seguintes resumem o significado dos sinais de controlo.

OE#	WE#	CS1#	CS2	operação
x	x	1	x	nenhuma
x	x	x	0	nenhuma
1	0	0	1	escrita
0	1	0	1	leitura

UB#	LB#	dados acedidos
0	0	16 bits
0	1	8 bits - maior peso
1	0	8 bits - menor peso
1	1	nenhum

   a) Desenhe o esquema de ligação.

   b) Sabendo que a duração mínima de WE# é 40 ns e que, na leitura, os dados são válidos 25 ns após o início de OE# e voltam a alta impedância 20 ns após o fim de OE#, calcule o número de estados de espera e de *data float*.

   c) Determine o valor a programar no *chip-select register* utilizado para a RAM, atribuindo o endereço base 0x0180 0000 (se não calculou os estados de espera e *data float*, arbitre e indique os valores).

2. [*2003/04v, 2º Teste*]
   Projecte uma placa de expansão de memória para o sistema AT91EB55, formada por 2 Mbyte de RAM e 2 Mbyte de Flash. Para a RAM, utilize *chips* K6X8008T2B-TF70, com a capacidade de 1 Mbyte, dispondo dos seguintes sinais de interface: dados, I/O1-8; endereços, A0-19; controlo, OE#, WE#, CS1#, CS2 ("#" significa *active low*). A tabela seguinte resume o significado dos sinais de controlo.

OE#	WE#	CS1#	CS2	operação	tempos de acesso
x	x	1	x	nenhuma	-
x	x	x	0	nenhuma	-
1	1	x	x	nenhuma	-
1	0	0	1	escrita	duração de WE# maior que 50 ns
0	1	0	1	leitura	dados válidos 70 ns após início de CE# alta impedância 25 ns após fim de OE#

Para a Flash, utilize um *chip* AT49BV1604-90, com a capacidade de 2 Mbyte organizados em $1M \times 16$, dispondo dos seguintes sinais de interface: dados, I/O0-15; endereços, A0-19; controlo, OE#, WE#, CE#. A tabela seguinte resume o significado dos sinais de controlo.

OE#	WE#	CE#	operação	tempos de acesso
x	x	1	nenhuma	-
1	1	x	nenhuma	-
1	0	0	escrita	duração de WE# maior que 100 ns
0	1	0	leitura	dados válidos 90 ns após início de CE# alta impedância 25 ns após fim de OE#

a) Desenhe o esquema de ligação, utilizando somente um sinal de *chip-select*.

b) Calcule, justificando, o número de estados de espera e de *data float* necessários para a RAM, para a Flash e para o conjunto.

c) Determine o valor a programar no *chip-select* register utilizado, atribuindo o endereço base 0x0240 0000 (se não calculou os estados de espera e *data float*, arbitre e indique os valores).

3. [*2004/05i, 1º Teste*]

a) Desenhe um bloco de memória com 1Mbyte usando memórias de 512Kbytes, organizadas em $512K \times 8 - bit$.

b) Ilustre o funcionamento da montagem apresentando a sequência de valores afectados a cada sinal ou barramento na execução das seguintes instruções:

```
strb r0, [r1]
stmdb r0, r1, r2, r3
```

Admita que os conteúdos de r0, r1, r2 e r3 são, respectivamente, 0x100, 0x110, 0x120 e 0x130.

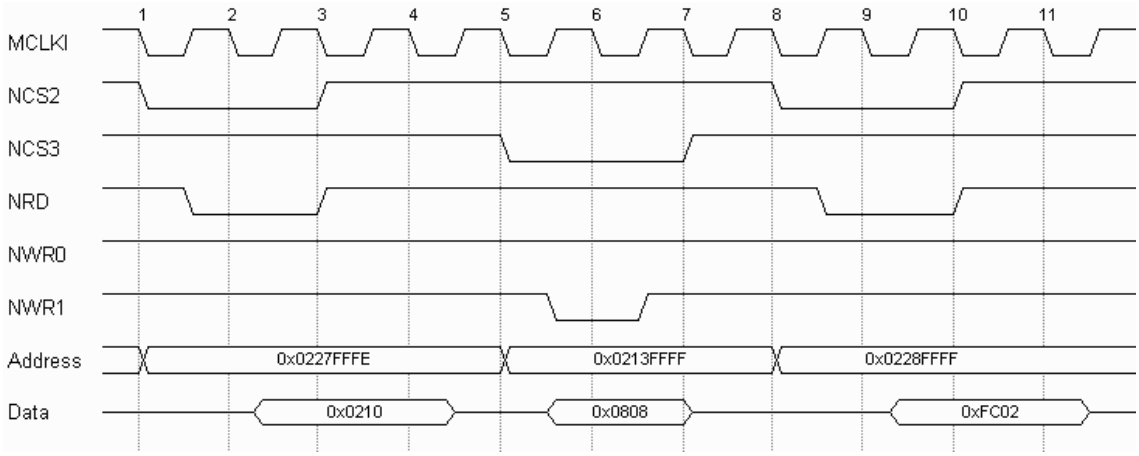
4. [*2006/07i, 1º Teste*]
A figura seguinte representa um *memory dump* de uma placa AT91EB55.

	0	4	8	C
0xffe00000	0x01002a5	0x02002221	0x03002b5	0x04002235
0xffe00010	0x02102221	0x02202221	0x00000000	0x00000000
0xffe00020	0x00000000	0x00000000	0x00000000	0x00000000

Relativamente a cada um dos registos de configuração do EBI, indique o conteúdo e descreva as características programadas.

5. [*2004/05v, 1º Teste*]
O diagrama temporal seguinte refere-se a um sistema AT91EB55 equipado com uma placa de expansão de memória com 2 Mbyte de RAM que utiliza os sinais de *chip select* NCS2 e NCS3.

Tendo em conta os sinais representados e sabendo que a placa de expansão é construída com circuitos de RAM de 512 Kbyte, organizados em $512K \times 8 - bit$,



a) Indique, justificando, quantos são os circuitos de RAM, de que forma estão ligados ao *bus* de dados e quais os endereços base atribuídos aos *chip-select* 2 e 3.

b) Identifique os diversos tipos de estados de espera expressos no diagrama, indicando quando ocorrem e o respectivo número de ciclos de *clock*.

c) Determine as palavras de programação correspondentes à configuração dos *chip-select* 2 e 3.

d) Desenhe o mapa de endereçamento, mencionando a FLASH e a RAM do sistema AT91EB55 e a expansão de memória. Considere que a FLASH e a RAM *onboard* estão localizadas nos endereços 0x01000000 e 0x02000000.

6. [*2004/05v, 2º Teste*]
Considere uma placa AT91EB55 e uma placa de expansão de RAM, formada por *chips* IC697MEM717, que têm os seguintes sinais de interface: dados, `I/O1-8`; endereços, `A0-17`; controlo, `OE#`, `WE#`, `CS1#`, `CS2` ("#" significa *active low*).

a) Baseando-se na tabela abaixo, descreva as características programadas nos registos de configuração dos chip-select `NCS2` e `NCS3`.

chip-select register 0	*chip-select register* 1	*chip-select register* 2	*chip-select register* 3
0x010022a5	0x02002221	0x02102225	0x02202225

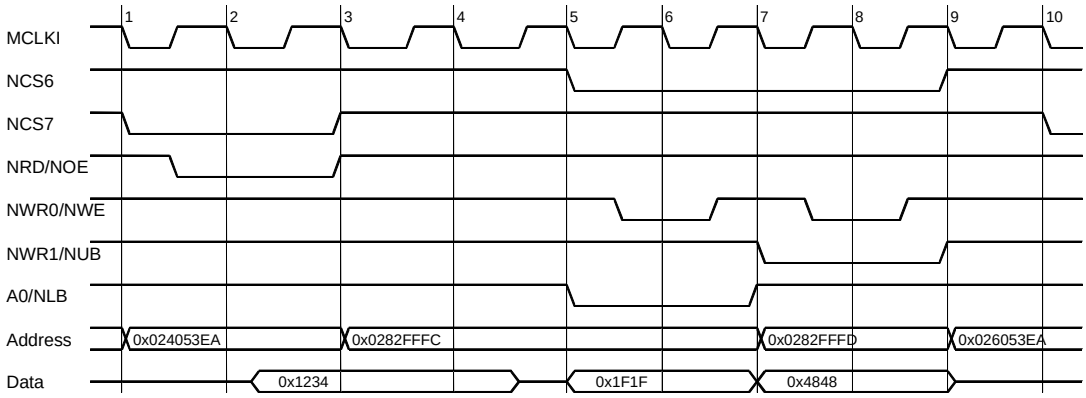
b) Desenhe o esquema de ligações da placa de expansão, admitindo que são usados quatro *chips* de RAM agrupados de modo a formar dois blocos de 512 kbyte, seleccionados pelos *chip-select* `NCS2` e `NCS3`.

c) Desenhe o mapa de endereçamento do sistema AT91EB55, incluindo a FLASH, a RAM original de 256 kbyte e a placa de expansão. Indique os endereços inicial e final, a dimensão e a ocorrência de *fold-back* para cada página de memória.

d) De modo a localizar toda a RAM (base e expansão) em endereços contíguos, indique as alterações a realizar ao esquema de ligações e ao conteúdo dos registos de configuração do EBI. Desenhe o mapa de memória resultante e indique o endereço base a utilizar no `ldscript` para localizar conteúdos na RAM.

7. [*2005/06i, 1º Teste*]
O seguinte diagrama temporal refere-se a uma placa de extensão de memória para o sistema AT91EB55, formada por dois *chips* de RAM com 2Mbyte, organizados em $1M \times 16 - bit$.

a) Indique o tipo e o número de estados de espera observáveis no diagrama, distinguindo os que são definidos pelo programador dos que são inseridos automaticamente.



b) Com base da informação expressa no diagrama, indique os valores programados em cada campo dos registos de configuração dos *chip selects* 6 e 7.

c) Indique a operação realizada nos ciclos de relógio 5 e 6, explicitando uma instrução assembly da arquitectura ARM que a origine.

d) Considere que no ciclo de relógio 10 se inicia a execução da instrução `ldrb r0,[r2]`. Qual o valor que fica em `r0`? Justifique.

8. [*2005/06i, 2º Teste*]
Pretende-se projectar uma placa de expansão que aumente em 4 Mbyte a capacidade de memória RAM do sistema AT91EB55. Para tal, dispõe-se de *chips* de RAM com capacidade de 2 Mbyte com os seguintes sinais de interface: `I/O`1-8, para dados; `A`0-20, para endereço; e `OE#`, `WE#` e `CS#`, para controlo ("#" significa active low). Estes *chips* de memória: *i)* requerem que o sinal de escrita se mantenha activo no mínimo por 40 ns; *ii)* disponibilizam os dados na sua saída 20 ns após a activação do sinal de leitura; e *iii)* necessitam, no máximo, de 35 ns para colocar o barramento em alta impedância após o fim de uma leitura. A memória deve ficar mapeada de modo a que seja possível ler e escrever no endereço 0x02700000.

a) Desenhe o esquema de ligações da placa de expansão.

b) Determine a palavra de configuração a programar no registo `nCS3`.

c) Desenhe o diagrama temporal referente à execução da instrução `strh r1,[r0]`, em que `r0`=0x02700250 e `r1`=0x20052006, representando a evolução de todos os sinais envolvidos.

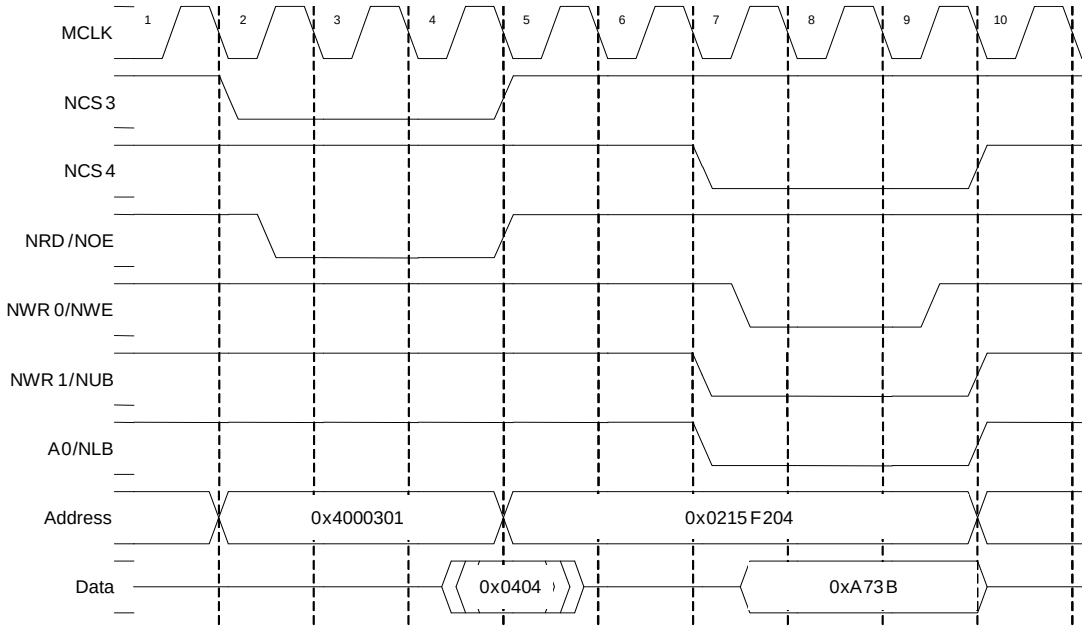
d) Indique as alterações a realizar ao esquema de ligações e à palavra de programação para se utilizarem memórias RAM com uma organização de $1M \times 16 - bit$, com sinais de controlo `OE#`, `WE#`, `UB#`, `LB#` e `CS#`. A placa de expansão mantém a capacidade de memória RAM, 4 Mbyte.

9. [*2005/06v, 1º Teste*]
O seguinte diagrama temporal refere-se a uma placa de extensão de memória para o sistema AT91EB55, formada por dois *chips* de RAM iguais com 1Mbyte, organizados em $512k \times 16 - bit$.

a) Com base na informação expressa no diagrama, complete a programação dos registos de configuração dos *chip selects* 3 e 4.

b) Indique uma instrução assembly da arquitectura ARM, envolvendo os registos `R0` e `R1`, que origine a operação realizada nos ciclos de relógio 7 a 9. Explicite os valores associados aos registos `R0` e `R1`.

10. [*2005/06v, 2º Teste*]
Considerando que o registo de configuração do sinal *chip select* 3 do EBI foi programado com a palavra 0x02103225,



a) Descreva as características programadas neste registo.

b) Tendo em conta a instrução `strb r0,[r1]`, indique o valor da palavra guardada no registo `r1` para que este enderece a posição 0x1FF do banco de memória controlado pelo sinal *chip select* 3.

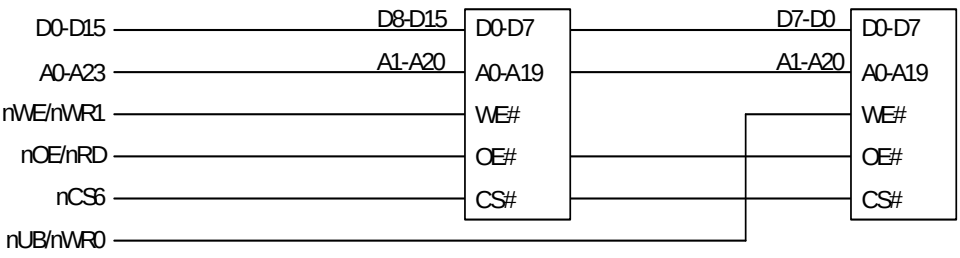
c) Desenhe o diagrama temporal referente à execução da instrução indicada na alínea anterior, em que `r0`=0x1A5F03DE, representando a evolução de todos os sinais envolvidos.

11. [*2005/06v, 3º Teste*]
Considere o esquema de ligações apresentado abaixo, referente a um módulo de expansão de RAM para o sistema AT91EB55. Os *chips* de memória utilizados disponibilizam os seguintes sinais para interface: `I/O0-7` para dados; `A0-18` para endereço; e `OE#`, `WE#` e `CS#`, para controlo ("#" significa active low). Estes *chips* de memória: *i)* requerem que o sinal de escrita se mantenha activo no mínimo por 48ns; *ii)* disponibilizam os dados na sua saída 24ns após a activação do sinal de leitura; e *iii)* necessitam, no máximo, de 31ns para colocar o barramento em alta impedância após uma leitura.

a) Determine a palavra de configuração a programar no registo `nCS5` para que seja possível ler e escrever no endereço 0x02500400.

b) Desenhe o diagrama temporal referente à execução da instrução `strh r1,[r0]`, em que `r0`=0x0252C3B4 e `r1`=0x2FA52DC4, representando a evolução de todos os sinais envolvidos.

12. [*2006/07i, 1º Teste*]
Considere o seguinte módulo de memória RAM ligado ao microcontrolador AT91M55800.



a) Qual a dimensão da memória representada no esquema anterior?

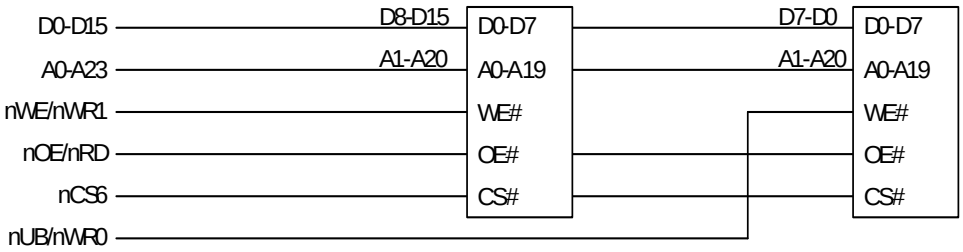
b) Qual o endereço base da memória se o valor do registo `EBI_CSR6` for 0x080022A1?

c) Qual a necessidade de existirem dois sinais de controlo de escrita?

13. [*2006/07i, 1º Teste*]
Descreva a acção efectuada por cada operação de escrita mencionada na função `buzzer_init`.

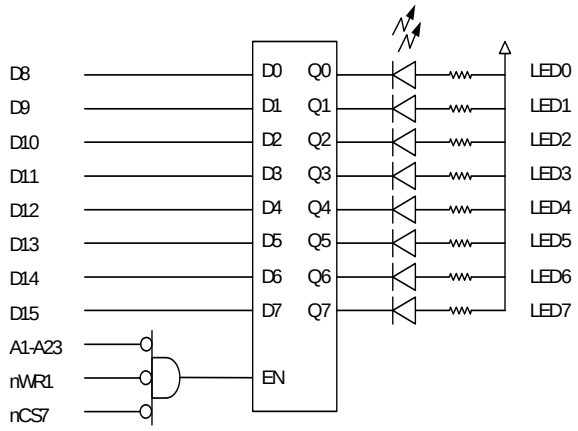
```
void buzzer_init()
{
        WRITE_U32(PIOA_BASE + PIO_PER, 1 << 14);
        WRITE_U32(PIOA_BASE + PIO_OER, 1 << 14);
        WRITE_U32(PIOA_BASE + PIO_CODR, 1 << 14);
}
```

14. [*2006/07i, 1º Teste*]
Para a utilização da seguinte montagem, 0x080020A1 é um valor de programação possível para o registo `EBI_CSR6`. Modifique o esquema apresentado de modo que o programador continue a ver o mesmo espaço de memória na mesma posição do espaço de endereçamento, se a programação for alterada para 0x080020A2.



15. [*2006/07i, 2º Teste*]
Considerando a seguinte montagem ligada ao sistema AT91EB55, programar as funções:

```
void light_set(int led, int state, int period);
void light_activity();
void light_init();
```



A função `light_set` coloca o LED, identificado pelo parâmetro `led` (LED0 a LED7), no estado definido no parâmetro `state`. Os estados possíveis são: apagado - valor 0, aceso - valor 1 ou a piscar - valor 2. Neste último caso o terceiro parâmetro indica o período em centenas de milisegundo.

A função `light_activity` serve para acender e apagar os LED no estado "piscar", de acordo com o ritmo definido na função `light_state`. Esta função não deve executar esperas, deve basear o seu processamento em variáveis de estado e num *timer* de *hardware*.

A função `light_init` faz as iniciações necessárias.

2.2 Timers/Counters

1. [*2006/07i, 1º Teste*]

   Considere o sistema de desenvolvimento estudado nas aulas.

   a) Implemente, em linguagem C, a função

   ```
   void InitTimer5(void);
   ```

   que inicia um *timer/counter* para permitir gerar um evento periódico, em intervalos de 250 ms.

   b) Dada a definição da seguinte estrutura de dados

   ```
   typedef struct {
   unsigned char sec; /* segundos */
   unsigned char min; /* minutos  */
   unsigned char hor; /* horas    */
   } relogio_t;
   ```

   e da variável global

   ```
   relogio_t relogio;
   ```

   escreva, em linguagem C, as funções

   ```
   void InitIntrTimer5(void);
   void IsrTimer5(void);
   ```

   A função `InitIntrTimer5` inicia o sistema de atendimento da interrupção. Admita a existência de uma rotina de encaminhamento de interrupção para o *timer/counter*, idêntica à estudada nas aulas, com o nome `IrqHandlerTimer5`.

   A função `IsrTimer5` realiza o atendimento da interrupção do *timer/counter* e actualiza o valor da variável global `relogio`, de forma a que esta represente um relógio com horas, minutos e segundos.

   c) Admita a existência da função `GetRelogio` com a seguinte definição:

   ```
   relogio_t GetRelogio(void)
   {
   relogio_t tmp;

   tmp.sec = relogio.sec;
   tmp.min = relogio.min;
   tmp.hor = relogio.hor;
   return tmp;
   }
   ```

   Comente a afirmação: *"Existe a possibilidade de a função retornar um valor inconsistente do relógio"*.

2.3 USART

1. [*2003/04v, 2º Teste*]
   Com base no código de manipulação da `USART`, que foi proposto nas aulas e realizado nos trabalhos, realize as alterações necessárias para implementar *byte-stuffing* de forma a evitar o aparecimento, na linha física, de caracteres com código inferior a 0x10. Para tal, cada byte de dados com valor inferior ou igual a 0x10 é substituído por uma sequência de dois bytes, o primeiro é o carácter `DLE` (0x10) e o segundo é o valor do byte de dados adicionado de 0x20.

a) Realize as alterações necessárias, implementando de novo os métodos `read` e `write`.

b) De forma a optimizar a gestão de espaço nos *buffers* de recepção e de transmissão, realize outra solução em que o *byte-stuffing* seja executado pela rotina de atendimento da interrupção.

2. [*2005/06v, 3º Teste*]
A estrutura interna de uma `USART` utiliza dois registos para a recepção (`RSR` e `RHR`) e dois registos para a transmissão (`TSR` e `THR`).

a) Explique para que serve cada um dos quatro registos.

b) Indique qual o estado dos registos que activa os sinais de `RxReady` e de `TxReady`.

c) Justifique a necessidade da existência de dois registos em cada sentido.

3. [*2007/08i, 2º Teste*]
Considere o sistema de desenvolvimento estudado nas aulas, baseado no microcontrolador AT91M55800.

a) Admita a seguinte declaração para o endereço base da USART 1

```
volatile U32 * const USART1 = (U32 *)0xfffc4000;
```

Explique, sucintamente, a importância do atributo `volatile` nesta declaração.

b) Implemente, em linguagem `C`, a função

```
void InitUsart1(void);
```

que inicia a USART 1 do sistema AT91M55800 de forma a possibilitar a comunicação série assíncrona no modo 19200 8N1.

c) Implemente, em linguagem `C`, a função

```
char Usart1GetChar(void);
```

que retorna o carácter recebido pela USART 1. No caso de não existir um carácter para retornar, a função deve ficar em espera pela chegada de um novo carácter.

d) Admita a seguinte implementação para a função `Usart1PutChar`, em que o endereço base da USART 1 é especificado pela declaração na alínea a):

```
void Usart1PutChar(char c)
{
USART1[0x1c/4] = c;
}
```

Indique, justificando, se existe alguma possibilidade de ocorrerem erros associados à transmissão de caracteres através da USART 1.

e) Comente a afirmação: *"No microcontrolador AT91M55800, a utilização de uma USART para comunicação série obriga sempre à utilização de um buffer."*

2.4 AIC

1. [*2004/05i, 1º Teste*]
O processamento de interrupções apresentado nas aulas inclui rotinas, escritas em assembly, que são os pontos de entrada de cada fonte de interrupção e chamam as respectivas funções de processamento, escritas em linguagem C.

a) Qual é a razão de existência destas rotinas?

b) Estas rotinas não guardam, em *stack*, os registos `R4` a `R10`. Porquê?

2. [*2004/05i, 1º Teste*]
Para que serve o mecanismo de prioridades do `AIC`?

3. [*2004/05i, 1º Teste*]
   Porque é que a posição de memória 0x18, referente à excepção `IRQ`, é preenchida com a instrução `ldr pc, [pc, #-0xF20]`?

4. [*2005/06i, 1º Teste*]
   Indique e justifique as vantagens da inclusão de um controlador de interrupções como o `AIC` num microcontrolador baseado na arquitectura ARM.

5. [*2007/08i, 2º Teste*]
   Relativamente ao processamento de interrupções no microcontrolador `AT91M55800A`,

   a) Apresente a motivação para a inclusão de um controlador de interrupções como o `AIC`.

   b) Indique as principais vantagens da duplicação dos registos de mascaramento de interrupções ao nível do processador, do `AIC` e dos periféricos internos ao microcontrolador.

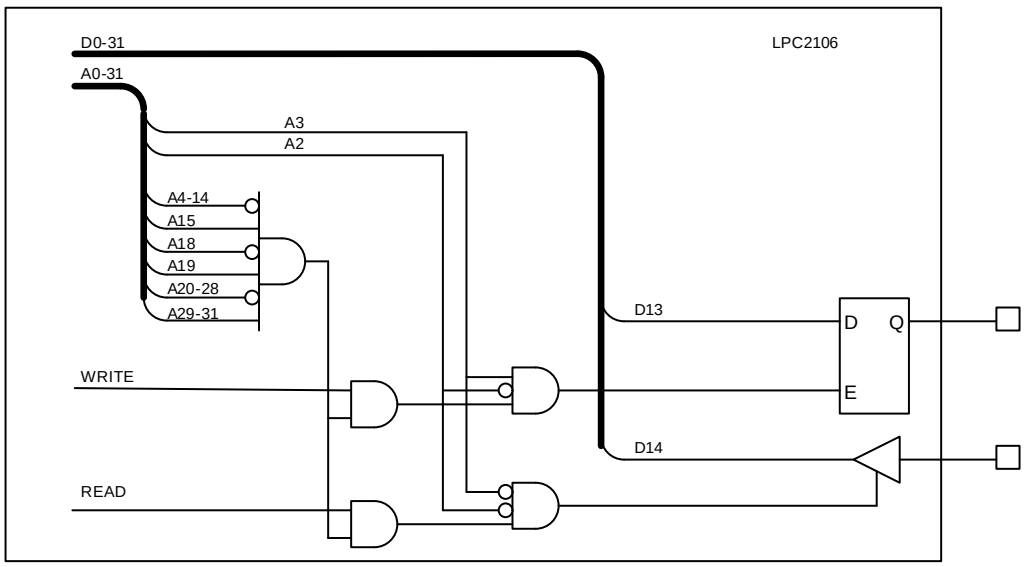
   c) Justifique as diferenças de funcionamento do registo de estado das `USART` (`CSR`) relativamente aos registos de estado dos `PIO` (`ISR`) e dos `TC` (`SR`).

Capítulo 3

Sistema de Desenvolvimento LPC2106

3.1 GPIO

1. [*2008/09i, 1º Teste*]
   Na figura seguinte está representado um periférico fictício pertencente ao microcontrolador LPC2106 que proporciona acesso a dois pinos externos.



   a) Programe as seguintes funções de acesso aos pinos externos:

```
void output(int value);
int input();
```

   A função `output` recebe no bit de menor peso do seu parâmetro o valor que deve ser colocado no pino de saída. A função `input` devolve um valor cujo bit de menor peso representa o valor lógico presente no pino de entrada.

   b) Faça um programa diferenciador que produza, no pino de saída, um impulso por cada transição do sinal aplicado ao pino de entrada. Os impulsos devem ter uma duração aproximada de 300 microsegundos.
   Utilize como base as funções anteriores e um timer do LPC2106.

2. [*2008/09i, 2º Teste*]
   No código C apresentado à frente os símbolos `LPC2106_BASE_GPIO` e `IOSET` representam,

respectivamente, o periférico `GPIO` e o seu registo interno `IOSET`. Para cada uma das situações de utilização seguintes, defina adequadamente estes símbolos.

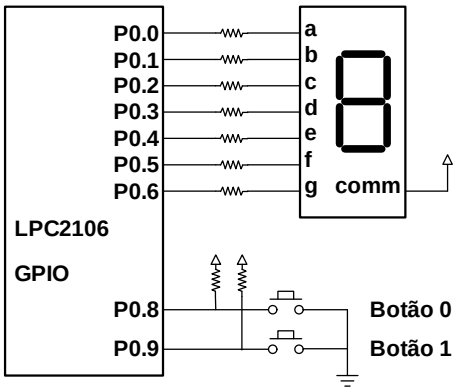
Nota: O endereço base do `GPIO` é `0xE0028000` e o registo `IOSET` tem um deslocamento de 4 *bytes* em relação à base.

a) `*((U32*)(LPC2106_BASE_GPIO + IOSET)) = 1 << 5;`

b) `LPC2106_BASE_GPIO[IOSET] = 1 << 5;`

c) `LPC2106_BASE_GPIO->IOSET = 1 << 5;`

3. [*2008/09i, 2º Teste*]
A figura seguinte representa a ligação de um *display* de 7 segmentos e dois botões de pressão ao microcontrolador LPC2106 através do `GPIO`.



a) Programe as seguintes funções de acesso aos pinos externos:

```
void display_init();
void button_init(int p);
```

A função `button_init` faz a iniciação do `GPIO` para permitir a leitura do estado de um botão de pressão, recebendo como parâmetro o pino do `GPIO` a que este está ligado. A função `display_init` faz a iniciação do `GPIO` para permitir interacção com o display de 7 segmentos.

b) Implemente, em linguagem C, a função

```
int button_read(int b);
```

que devolve o estado do botão de pressão ligado ao pino `p` do `GPIO`. Assuma que o valor 1 corresponde ao estado botão pressionado e o valor 0 ao estado botão não pressionado. Ignore o efeito de *bounce*.

c) Escreva, em linguagem C, a função

```
void display_write(int v);
```

que mostra no *display* de 7 segmentos a representação hexadecimal do valor que lhe é passado como parâmetro.

d) Construa um programa, em linguagem C, que utilize a montagem anterior para implementar um contador ascendente/descendente de módulo 16. Nesta aplicação, o pressionar do botão 0 coloca o contador no modo ascendente, enquanto que o pressionar do botão 1 altera o modo do contador para descendente. O valor da contagem, visível no *display*, é actualizado automaticamente com uma cadência de 1 segundo. Use um *timer* como referência de tempo.

e) Reescreva a função `button_read` filtrando o efeito de *bounce*.

3.2 Relógio de Tempo Real (RTC)

1. [*2008/09i, 1º Teste*]
   Relativamente ao relógio de tempo real (RTC) do microcontrolador LPC2106:

   a) Explique porque é que uma consulta de data e hora obriga a efectuar, no mínimo, duas leituras aos registos.

   b) Qual a motivação para a existência dos registos compactados `CTIME0`, `CTIME1` e `CTIME2`.

3.3 Memória FLASH

1. [*2008/09i, 2º Teste*]
   Programe a função `flash_write` para escrita de um bloco de dados na memória FLASH do LPC2106. O parâmetro `flash_address` representa o endereço da FLASH onde se pretende escrever e considera-se que coincide com o início de um bloco de FLASH. O parâmetro `address` representa um endereço da RAM de onde se vão copiar os dados e size representa a quantidade de dados a escrever. Considera-se que `size` é múltiplo da dimensão do bloco de FLASH.

```
size_t flash_write(int flash_address, void * address, size_t size);
```

Capítulo 4

Organização de Software para Sistemas Embebidos

1. [*2004/05v, 1º Teste*]
   Com base no código de manipulação da USART, que foi proposto nas aulas e realizado nos trabalhos, pretende-se adicionar à classe `Usart` os seguintes métodos

   ```
   class Usart: Interrupt{
           ...
           public:
           ...
           bool write(U8 data, U32 timeout);
           bool read(U8 * data, U32 timeout);
   };
   ```

   O método `write` entrega um byte para ser enviado; se a fila de transmissão estiver cheia, espera `timeout` milissegundos que haja espaço disponível; retorna `true` se terminar com sucesso.

   O método `read` preenche `data` com um byte recebido; se a fila de recepção estiver vazia, espera `timeout` milissegundos que haja dados; retorna `true` se terminar com sucesso.

   a) Admitindo a existência da variável global `clock`, que representa a contagem de tempo, em milissegundos, escreva os métodos `write` e `read`.

   b) Escreva uma classe para gerir o *timer* 2, com métodos para iniciar o seu funcionamento e processar a interrupção, incrementando a variável `clock` a intervalos de 1 milissegundo.

2. [*2004/05i, 1º Teste*]
   A figura seguinte representa uma ligação de um teclado de matriz ao EB55.

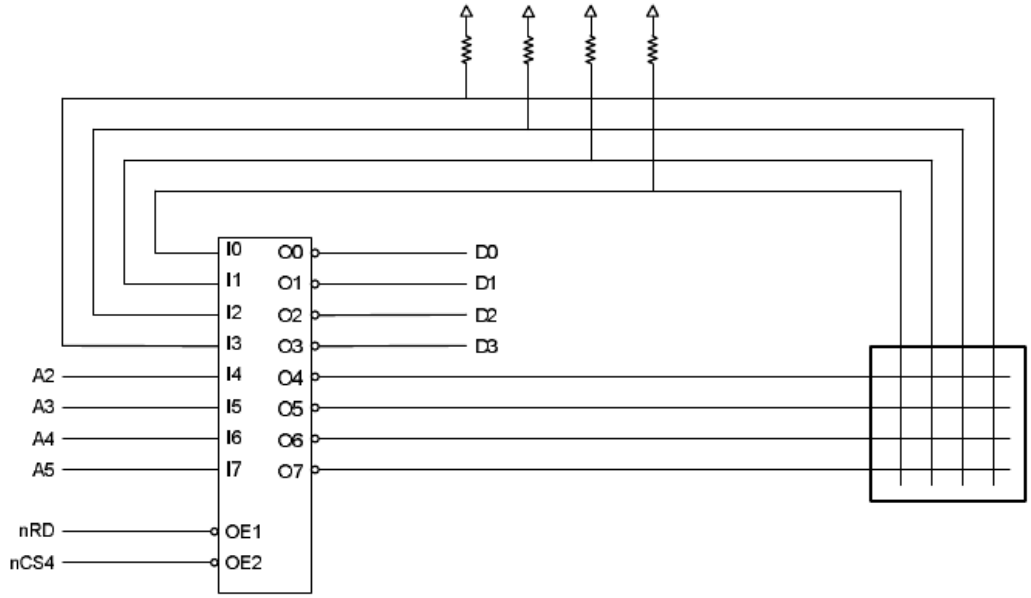
   a) Descreva sucintamente o princípio de funcionamento.

   b) Programe uma função de leitura do teclado com o seguinte funcionamento:
   Permanece em espera enquanto não houver tecla premida.
   Quando uma tecla for premida devolve imediatamente o seu código.
   No caso de uma segunda chamada, permanece em espera enquanto se mantiver premida a tecla anterior.

   c) Apresente uma solução que permita detectar e armazenar códigos de teclas sem que o programa principal tenha chamado a função de leitura.

3. [*2004/05i, 2º Teste*]
   Escreva o código da classe `Adc` com a interface apresentada. Esta classe destina-se a controlar o ADC na digitalização de um sinal.

   As funções `start` e `stop` definem os momentos de início e fim da digitalização.



A função `init` executa as inicializações necessárias e define o ritmo de conversão (amostras por segundo).

A função `read` copia amostras para a memória apontada por `buffer` até atingir a totalidade dos dados armazenados ou a capacidade da memória, indicada por `size`. Retorna o número de amostras copiadas.

```
class Adc {
public:
        void init(int rhythm);
        void start();
        void stop();
        size_t read(U8 * buffer, size_t size);
};
```

Uma utilização possível desta classe é a seguinte:

```
Adc adc;
U8 buffer[];
int main() {
        ...
        adc.init(8000);
        adc.start();
        while (...) {
                ...
                adc.read(buffer, sizeof(buffer));
                ...
        }
        adc.stop();
        ...
        adc.read(buffer, sizeof(buffer));
        ...
}
```

4. [*2006/07i, 1º Teste*]
   Admita a possibilidade de o código RC5 ser transmitido com *bit time* diferentes de 1,8 ms. Faça uma função que determine o *bit time* para uma dada transmissão.

5. [*2004/05v, 1º Teste*]
   Admitindo que dispõe de um emissor de infravermelhos controlado pelo pino `PB2` do sis-

tema AT91EB55, escreva uma classe para transmitir dados, utilizando o protocolo Philips RC5, com a seguinte interface:

```
class TxRC5{
        ...
public:
        void init();
        void write(int adr, int cmd);
};
```

A função `init` realiza as iniciações, incluindo a configuração do `PIO`, *timer* e interrupções.

A função `write` promove o envio de uma trama cujos campos `address` e `command` contêm os valores indicados pelos parâmetros `adr` e `cmd`. O bit de `toggle` deve ser produzido automaticamente.

6. [*2004/05v, 2º Teste*]

   a) Escreva a classe `LedShifter` com a interface seguinte:

   ```
   class LedShifter{
           ...
   public:
           void init();
           void shift();
   };
   ```

   Esta classe implementa um registo de deslocamento em que se faz deslocar um LED aceso sobre o conjunto de LEDs da placa AT91EB55.

   O método `init` serve para fazer as iniciações.

   O deslocamento do LED ocorre quando for executado o método `shift`. O sentido é de `D1` para `D8`; após o último, recomeça do primeiro.

   b) Escreva a classe Timer com a interface seguinte:

   ```
   class Timer{
           ...
   public:
           void init();
           void wait(int units);
   };
   ```

   Esta classe controla um *timer/counter* de modo a gerar uma interrupção periódica a intervalos de 100 ms, fazendo uma contagem interna dos períodos decorridos.

   O método `init` serve para fazer as iniciações de *hardware* e iniciar a contagem de períodos.

   A aplicação executa `wait` para esperar que decorra um número de períodos indicado pelo parâmetro `units`. Esta acção consome os períodos indicados, subtraindo-os à contagem acumulada.

   c) Escreva o código de uma aplicação que usa as classes `LedShifter` e `Timer` para deslocar o LED aceso a cada 300 ms.

   d) Relativamente ao código das alíneas anteriores, refira o que seria necessário acrescentar ou alterar, para que a aplicação pudesse permanecer sensível aos botões de pressão `SW3` e `SW4`. Estes botões seriam usados para incrementar e decrementar o número de períodos entre deslocamentos.

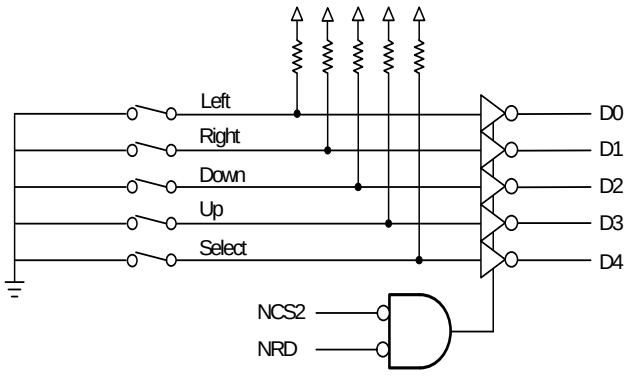
7. [*2005/06i, 1º Teste*]
   Pretende-se construir um sistema de controlo de qualidade para uma linha de montagem, em que os objectos são deslocados num tapete rolante com velocidade constante. Para a monitorização é usado um sensor que detecta a passagem dos objectos. Uma vez que a velocidade do tapete é fixa, a resposta do sensor é directamente proporcional à dimensão do

objecto, tendo sido determinado que os objectos aceitáveis originam um estímulo no sensor (saída activa com o valor lógico '1') com duração entre 0,6 e 0,8 s. O sistema sinaliza a detecção de objectos defeituosos activando uma luz vermelha, que é apagada quando passa um objecto aceitável ou quando é pressionado um botão de controlo.

a) Descreva sucintamente uma solução baseada no sistema AT91EB55.

b) Apresente o código que implementa a solução descrita na alínea anterior.

8. [*2005/06i, 2º Teste*]
A figura abaixo representa a ligação de um conjunto de botões de navegação ao sistema AT91EB55, mapeado sobre o endereço de memória 0x06000000.



Este periférico é representado em C++ por uma classe com a seguinte interface pública:

```
class NavigationKeys {
        ...
        public:
        enum Event { NONE, LEFT, RIGHT, DOWN, UP, SELECT };
        ...
        public:
        Event getEvent();
        ...
};
```

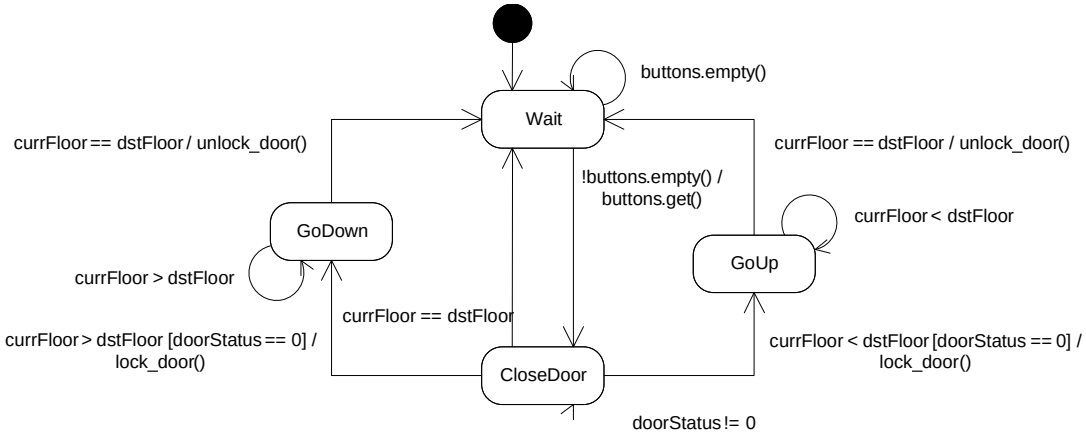
em que o método `getEvent` retorna `LEFT`, `RIGHT`, `DOWN`, `UP` ou `SELECT` sempre que o botão correspondente está pressionado e tenha havido alteração do estado desse botão desde a última chamada ao método. Em todos os outros casos, `getEvent` retorna `NONE`.

a) Apresente o código do método `getEvent` para uma solução baseada na pesquisa de estado dos botões. Ignore o problema do *bounce*.

b) Modifique a classe para memorizar os eventos associados aos botões de navegação. Nesta situação, a invocação do método `getEvent` retorna o evento mais antigo em memória.

c) Apresente as vantagens e desvantagens das duas soluções propostas, indicando exemplos de situações em que cada uma pode ser aplicada.

9. [*2005/06v, 1º Teste*]
Pretende-se construir um sistema de controlo para um elevador que será instalado num prédio com três pisos. Para accionar o elevador existe uma botoneira com três botões instalada dentro da sua cabina, correspondentes aos botões de R/C, 1ºe 2ºandar, e um botão de chamada em cada piso ligado em paralelo com o botão da botoneira correspondente a esse piso. O sistema disponibiliza duas saídas para controlo do sentido do motor do elevador (`moveUp` e `moveDown`) e outra para controlo da abertura da porta da cabina (`doorLocked`), para além de duas entradas que indicam o piso em que o elevador se encontra em cada instante (`currFloor`) e o estado da porta da cabina do elevador (`doorStatus`). O diagrama de estados do sistema é o seguinte.



Considerando que, no protótipo do sistema, a botoneira da cabina do elevador é implementada no sistema AT91EB55 pelos botões de pressão `SW1` a `SW3`, que as três saídas do sistema de controlo são apresentadas nos LEDs `D1` a `D3` e que as suas duas entradas podem ser lidas nos endereços 0x02204C80 (`currFloor`) e 0x02204C84 (`doorStatus`),

a) Escreva a classe `Botoneira` que implementa uma interface com armazenamento de eventos para os botões `SW1` a `SW3`.

b) Apresente o código correspondente à máquina de estados descrita no diagrama anterior.

c) Modifique a implementação para que o sistema permaneça no estado `Wait` por, pelo menos, 5s aquando de uma transição dos estados `GoDown` ou `GoUp`.

10. [*2005/06v, 2º Teste*]
Pretende-se construir um sistema de controlo para uma passagem de nível sobre uma linha férrea única (a mesma linha serve os dois sentidos) que actue sobre as cancelas e as sinalizações luminosa e sonora. Este sistema dispõe de 2 detectores de presença de comboio, localizados a 3km da passagem de nível, para detectar a entrada e saída de comboios na zona controlada pelo sistema. A saída de cada sensor fica activa enquanto um comboio o atravessa. Aquando da aproximação de um comboio, o sistema de controlo deve activar os sinais sonoro e luminoso e, passados 10s, fazer baixar as cancelas. A posição final de cada cancela é sinalizada pela activação de um sensor de fim de curso próprio.
Após a passagem do comboio, ou seja, quando o comboio tiver cruzado o detector de proximidade localizado no extremo oposto da passagem de nível, o sistema levanta as cancelas, desactivando os sinais luminosos e sonoro quando estas atingirem o final do seu curso.

a) Descreva sucintamente uma solução baseada no sistema AT91EB55.

b) Apresente o código que implementa a solução descrita na alínea anterior.

11. [*2005/06v, 3º Teste*]
Pretende-se monitorizar o número de unidades produzidas numa linha de montagem de uma unidade fabril. Este controlo será realizado à custa de um sistema AT91EB55 e de um sensor que detecta a passagem de objectos num tapete rolante. O sensor é baseado num feixe luminoso que, quando interrompido pela passagem de um objecto no tapete, provoca a activação da sua saída. O sistema de controlo incrementa um contador interno de unidades, de 16 bits, sempre que é detectada a passagem de um objecto no tapete.
Para monitorização do volume de produção, o sistema de controlo aceita pedidos por um canal série com características 14400 8E2. Ao receber o carácter 'N', o sistema transmite o valor do contador interno de unidades.

a) Descreva sucintamente uma solução baseada no sistema AT91EB55.

b) Apresente o código que implementa a solução descrita na alínea anterior.

12. [*2006/07i, 1º Teste*]
Considere um dispositivo de controlo de uma porta com a seguinte composição: um botão de pressão, dois actuadores, um para fechar e outro para abrir, e dois sensores de posição, um de porta aberta e outro de porta fechada.

O comportamento do dispositivo de controlo deve ser o seguinte: quando se carrega no botão a porta pára, se estiver em movimento, ou inicia um movimento, contrário ao anterior, mas só se estiver parada há mais de 3 segundos.

a) Desenhe a máquina de estados. Recomenda-se que enumere os objectos intervenientes (botão, actuadores, etc) e as respectivas operações, e que defina os eventos.

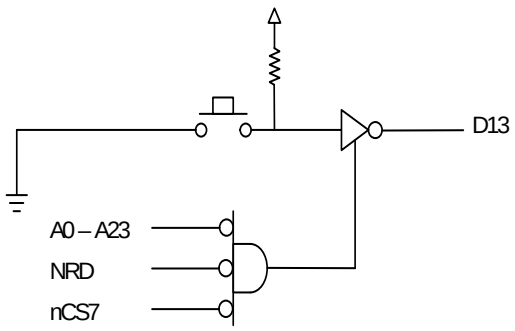
b) Desenhe o esquema de ligações com vista à realização deste dispositivo com base no AT91EB55.

c) Programe a função de leitura de eventos.

d) Programe a máquina de estados.

13. [*2006/07i, 1º Teste*]
Considerando a seguinte montagem, programe as funções `button_state` e `button_transition` tendo em consideração que o botão produz *bounce* com 30 milisegundos de duração.

```
int button_state();
int button_transition();
```



A função `button_state` devolve o estado actual do botão: 0 se estiver premido ou 1 se estiver largado. A função `button_transition` devolve 0 se não ocorreu mudança de estado do botão desde a chamada anterior, devolve 1 se foi premido depois da chamada anterior ou 2 se foi largado depois da chamada anterior.

14. [*2006/07i, 2º Teste*]
Um sistema de controlo de um vidro de automóvel é constituído por: dois actuadores de movimento, um de abertura (`AA`) e um de fecho (`AF`); dois botões de pressão, um para indicar a abertura (`BA`) e outro para indicar o fecho (`BF`); dois sensores de fim de curso, um de vidro aberto (`SA`) e outro de vidro fechado (`SF`); um sensor de segurança que detecta, durante o fecho do vidro, a existência de um obstáculo.

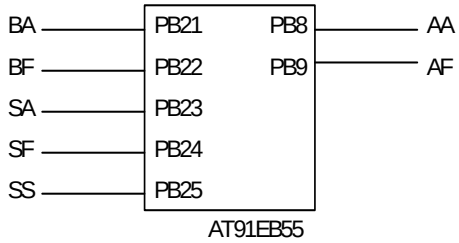
Quando um botão é premido o vidro inicia imediatamente o movimento correspondente. Se o botão se mantiver premido por mais de 3 segundos o vidro continua o movimento até ao fim do curso mesmo que o botão seja largado.

Se durante o movimento de fecho o sensor de segurança for activado o controlo inverte imediatamente o movimento do vidro.

Considere o seguinte esquema de ligações e que dispõe das funções `timer_init`, `timer_start` e `timer_timeout` estudadas nas aulas.

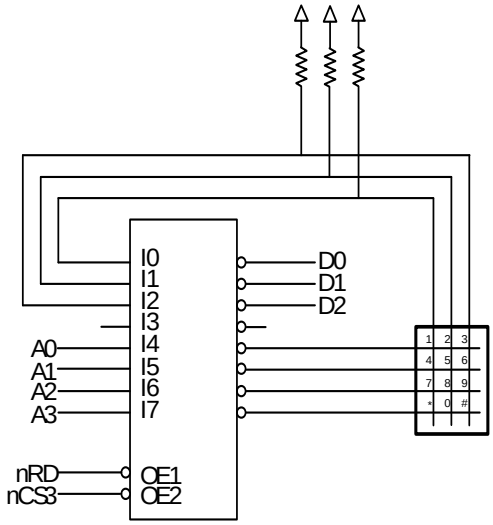
a) Desenhe a máquina de estados. Recomenda-se que defina e caracterize bem os eventos.

b) Programe a máquina de estados e a geração dos eventos.



15. [*2007/08i, 2º Teste*]
A figura 1 representa a ligação de um teclado em matriz espacial de 3×4 posições ao sistema AT91EB55. O teclado é acessível através da `EBI` do processador a partir do endereço `0x05100000` e de um *buffer tri-state* com as seguintes características aquando da activação dos sinais $\overline{OE1}$ e $\overline{OE2}$:

- $t_{minZ \to H} = 15ns$;
- $t_{minZ \to L} = 27ns$;
- $t_{maxH \to Z, L \to Z} = 20ns$.



a) Determine a palavra de configuração a programar no registo `nCS3`.

b) Considerando uma solução baseada na pesquisa de estado do teclado, implemente, em linguagem `C`, a função

```
int getKey(void);
```

que devolve o valor numérico correspondente à tecla que está a ser pressionada. Admita que apenas uma tecla é pressionada em cada instante de tempo.

c) Admitindo a existência de um *ring buffer* `buff_keyb` com `DIM` posições e com a seguinte interface

```
int isFull(void);   \* Devolve 1 se o buffer esta cheio, 0 no caso contrario *\
int isEmpty(void);  \* Devolve 1 se o buffer esta vazio, 0 no caso contrario *\
void put(int val);  \* Insere um valor no buffer *\
int get(void);      \* Retira e devolve  valor mais antigo do buffer *\
```

escreva, em linguagem `C`, a função

```
void isrTimer(void);
```

que realiza o atendimento da interrupção de um *timer/counter* e memoriza em `buff_keyb` o valor numérico correspondente à tecla que está a ser pressionada.

d) Modifique a função `getKey` para retornar a tecla mais antiga armazenada em `buff_keyb`.

e) Considere a seguinte definição para a função `get` da interface com o *ring buffer* `buff_keyb`:

```
int get(void)
{
  int val = buff_keyb[idx_get++];
  if(idx_get == DIM)
    idx_get = 0;
  --size;
  return val;
}
```

Comente a afirmação: *"Existe a possibilidade de a função originar corrupção dos dados armazenados no buffer."*