Parte II - Sistemas Digitais de Aplicação Genérica

Nos projectos do capítulo anterior, os circuitos são implementados de forma a suportarem a execução de uma operação ou funcionalidade. Por exemplo, o circuito do multiplicação apenas executa multiplicações. No caso de pretendermos executar uma outra operação ou funcionalidade teremos, muito provavelmente, de desenvolver um outro circuito. Dizemos que são *circuitos de aplicação específica ou dedicados*. É claro que um determinado circuito dedicado pode realizar mais de uma funcionalidade. No entanto, nestes casos, toda a funcionalidade que se pretende executar com o circuito terá de ser tida em conta na fase de projecto.

Sendo circuitos dedicados à execução de uma funcionalidade, significa que estão optimizados para tal e, consequentemente, apresentam um bom desempenho. Contudo, caracterizam-se por serem pouco flexíveis e não suportarem a execução genérica de aplicações.

Se o objectivo é ter um sistema que permita a execução genérica de aplicações, então devemos procurar uma arquitectura que permita a execução genérica de aplicações. Pretendemos, neste caso, um circuito de aplicação genérica.

Para chegar a uma arquitectura deste tipo, podemos extrapolar o que foi feito no capítulo anterior e considerar que de uma forma genérica são necessários registos, operadores e encaminhadores de dados (*multiplexers*). A diferença desta nova arquitectura relativamente a uma de aplicação dedicada é que em vez de associar uma operação específica a um determinado registo, vamos considerar que temos um conjunto de registos (e.g., na forma de banco de registos) a que podemos aplicar operações com recurso a uma ALU capaz de executar operações aritméticas e lógicas típicas (somador, subtractor, multiplicador, etc.).

A arquitectura do nosso sistema de aplicação genérica será assim formado por uma ALU e por um banco de registos. A interligação entre a ALU e o banco de registos pode ser feita de diversas formas, mas neste caso consideremos que a ALU recebe dois operandos do banco de registos e envia o resultado para o banco de registos. Adicionalmente, vamos admitir uma implementação combinatória para a ALU, permitindo que cada operação de transferência entre registos ocorra num único ciclo de relógio.

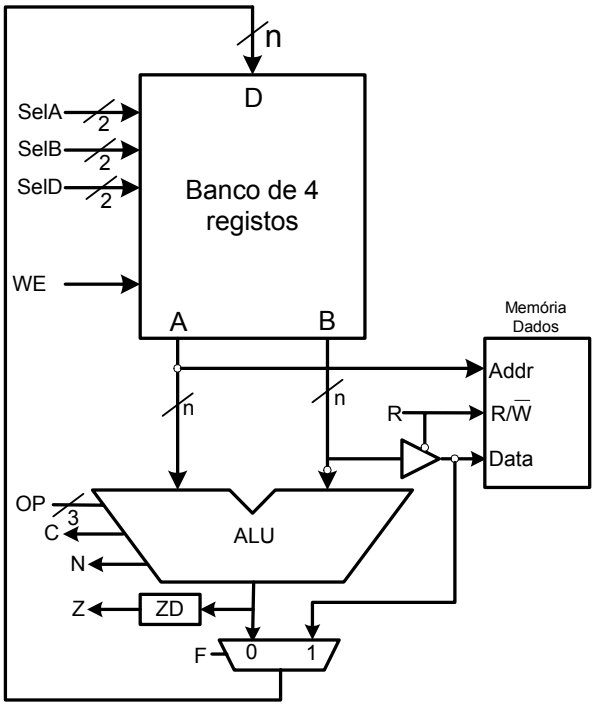
Para completar a arquitectura, vamos ainda incluir uma memória de onde podemos ler e escrever dados de e para os registos, respectivamente. Quer os dados, quer os endereços de memória provêm de registos do banco de registos (ver figura 14).

No circuito da figura, considerou-se um banco de quatro registos e uma ALU com oito operações (OP com três bits) identificados de acordo com a tabela ao lado. A ALU inclui ainda três bits de estado, C, N, e Z, que indicam o valor do *carry*, do sinal do resultado e se o resultado é igual ou diferente de zero.

A escrita na memória utiliza como endereço o valor que se encontra na saída A do banco de registos e como dados o valor que se encontra na saída B do mesmo banco. A leitura da memória também utiliza como endereço o valor que se encontra na saída A do banco de registos e os

dados lidos da memória são enviados para o banco de registos através do *multiplexer* à saída da ALU.

Para exemplificar a utilização da arquitectura, recorramos de novo ao problema de multiplicação com somas sucessivas. Para simplificar, consideremos que o multiplicando X e o multiplicador Y se encontram nas posições de memória 0 e 1, respectivamente, que os registos estão todos a 0 e que o resultado final deve ficar na posição 2 da memória.



OP	Operação
000	O = A
001	O = A + B
010	O = A - B
011	O = A + 1
100	O = A – 1
101	O = A × B
110	O = A OR B
111	O = NOT A

Figura 14 – Exemplo de um circuito de aplicação genérica

Uma vez que a unidade de processamento já está desenvolvida[1], é necessário apenas projectar a unidade de controlo. Para tal, comecemos por determinar a sequência de acções da unidade de controlo tendo em conta a unidade de processamento proposta (ver figura 15).

Inicialmente, de acordo com o fluxograma do lado esquerdo da figura 15, após a activação do sinal *início*, realizam-se os passos seguintes para carregar os operandos X e Y:

- Lê-se o operando X da memória com o endereço dado pelo registo R0 e guarda-se no registo R1;
- Incrementa-se o registo R0;
- Lê-se o operando Y da memória com o endereço dado pelo registo R0 e guarda-se no registo R2.

De seguida, testa-se o conteúdo do registo R2 (operando Y). Se for 0, segue-se o armazenamento do resultado na memória. Caso contrário, acumula o multiplicando (operando X guardado em R1) em R3 e decrementa o multiplicador (operando Y guardado em R2). Esta sequência de acções repete-se até que R2 seja 0.

[1] Repare que neste caso, a arquitectura foi proposta antes do problema. Ao contrário das arquitecturas dedicadas em que se tem de conhecer o problema antes de propor uma arquitectura.

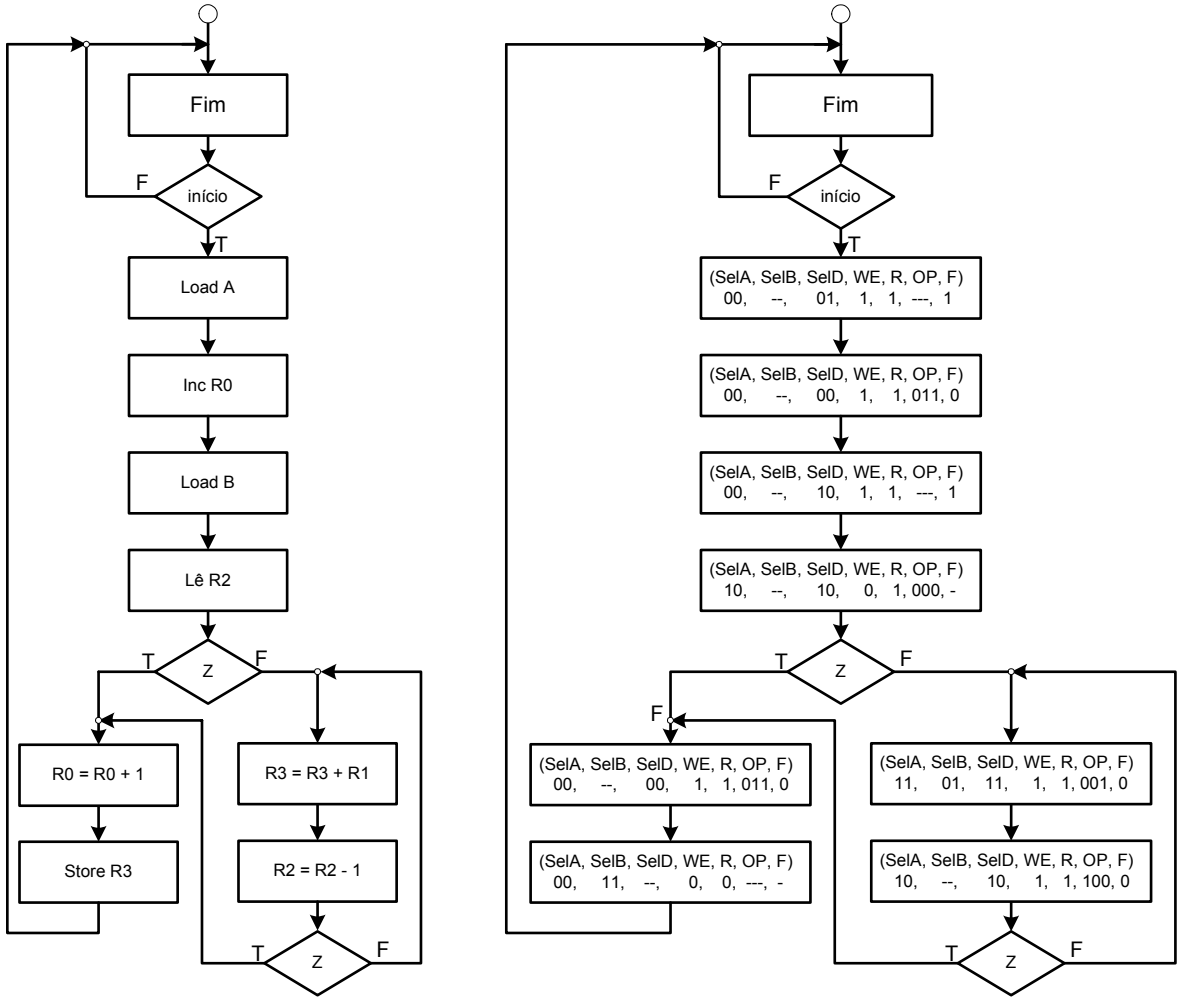


Figura 15 – Especificação da unidade de controlo para a multiplicação por somas sucessivas

A escrita do resultado na memória é feita no endereço 2. Para tal, é necessário incrementar R0, que tem 1 (último endereço lido) e depois escrever o valor de R3 na memória.

Para executar cada uma destas acções, é necessário actuar nos sinais de controlo da unidade de processamento (Sela, SelB, SelD, WE, R, OP, F – palavra de controlo) de acordo com o que se pretende e tendo em conta a funcionalidade de cada um dos módulos do diagrama de blocos. Por exemplo, no início, para carregar o operando A é necessário:

- Seleccionar o registo R0 (contém o endereço de memória onde se encontra o operando X) de forma a aparecer na saída A do banco de registos (usada como endereço de memória);
- Seleccionar o registo R1 como destino do operando lido da memória;
- Activar a escrita no banco de registos, com WE = 1;
- Activar o bit de controlo R/W = 1 para leitura da memória;
- Seleccionar a entrada 1 (F = 1) do *multiplexer* de saída da ALU.

Através do exemplo, verificamos que cada uma das acções pode ser vista como uma instrução dada pelo controlador à unidade de processamento para realizar uma determinada operação. Se identificarmos cada uma das instruções com uma mnemónica, teríamos, como exemplo, as instruções seguintes:

load R1 – carrega o registo R1 com o conteúdo da memória no endereço dado por R0
store R2 – carrega em R o conteúdo da memória no endereço dado por R2
inc R0 – incrementa o conteúdo do registo R0
dec R2 – decrementa o conteúdo do registo R2
add R3, R1 – soma o conteúdo de R3 com o de R1 e escreve o resultado em R3
jmpz R2, instx – executa outra sequência de instruções a começar em *instx* caso R2 = 0
jmpnz R2, instx – executa outra sequência de instruções a começar em *instx* caso R2 ≠ 0

No caso de pretendermos executar um outro algoritmo baseado nas instruções descritas, basta gerar uma outra unidade de controlo com uma nova sequência de acções que cumpra a funcionalidade do algoritmo.

Nesta altura, ainda não podemos falar de um sistema de aplicação genérica pois, apesar de temos uma unidade de processamento genérica que suporta a execução genérica de algoritmos (desde que possam ser descritos como uma sequência de instruções que podem ser executadas na unidade de processamento), a unidade de controlo não é genérica pois depende do algoritmo, ou seja, da sequência de instruções pretendida. Para que se consiga um sistema de aplicação genérica, é necessário um controlador programável, isto é, um controlador que possa ser programado com uma determinada sequência de acções sem alterar a estrutura hardware. A este tipo de controlador designamos *controlador programável*.

Uma solução de implementação de um controlador programável baseia-se no uso de uma memória de instruções onde é guardada a sequência de instruções que se pretende executar. Depois, o controlador apenas tem de aceder à memória, ler sequencialmente cada uma das instruções e executá-las na unidade de processamento. Em cada momento, o endereço de memória que contém a próxima instrução a ser lida é mantido num registo específico designado *Program Counter* (PC). Por defeito, o PC é incrementado em uma unidade ao longo da execução do algoritmo. No entanto, como acontecia com o algoritmo de multiplicação, esta sequência pode ser alterada devido a um teste de uma variável. Como tal, o controlador deve permitir carregar o PC com um valor diferente de PC + 1 (ver figura 16).

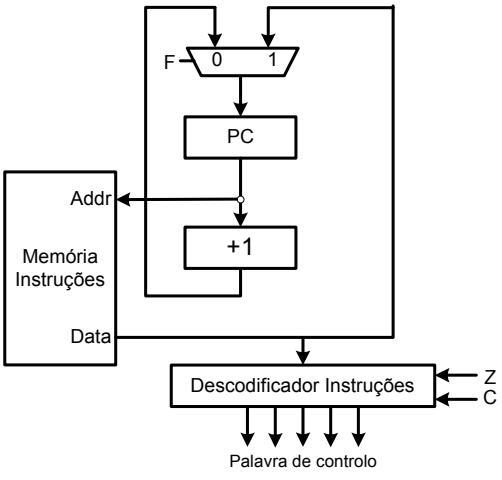


Figura 16 – Unidade de controlo programável

O registo PC contém o endereço da primeira instrução a executar. Em seguida, após a instrução ser lida e executada, é incrementado para que fique a apontar para a próxima instrução. Esta sequência pode ser alterada caso surja uma instrução de alteração da sequência. Este tipo de instruções testa um determinado bit de estado e caso se verifique a condição, força o controlador a carregar o PC com um determinado valor indicado na instrução (entrada 1 do *multiplexer* à entrada do PC), fomentando, desta forma, um "salto" para outro endereço da memória de instruções. Nesta situação, o PC é carregado com o novo endereço.

Retomando o exemplo do multiplicador, e considerando as instruções mencionadas atrás, teríamos a sequência de instruções seguinte:

Endereço	**Instrução**
0	load R1
1	inc R0
2	load R2
3	jmpz 7
4	add R3, R1
5	dec R2
6	jmpnz 4
7	inc R0
8	store R3

Na implementação, a cada uma destas instruções corresponde uma sequência binária que identifica o tipo de instrução e os seus parâmetros. É a partir desta sequência que o controlador gera a palavra de controlo através de um bloco designado *descodificador de instruções*. Este é um circuito combinatório que tem como entrada o código da instrução, juntamente com os seus parâmetros ou operandos, os bits de estado da unidade de processamento e gera a palavra de controlo. O descodificador pode ser implementado com uma ROM (naturalmente, pois uma ROM permite implementar funções lógicas) que recebe como endereço o código da instrução e os bits de estado e coloca na saída de dados a palavra de controlo (uma descrição prática da geração do descodificador de instruções será considerada no capítulo seguinte).

A unidade de controlo programável, juntamente com a unidade de processamento genérica, formam um circuito de aplicação genérica designado *Unidade Central de Processamento* (CPU). No projecto de um CPU, é necessário especificar o conjunto de instruções que deve suportar, a codificação das instruções, a unidade de processamento e a unidade de controlo, e em particular o descodificador de instruções.