Circuitos Sequenciais

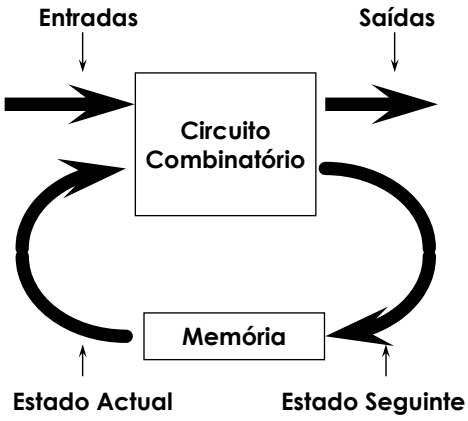
Escola Naval - Dep. Armas e Electrónica

v1.5 2005

CIRCUITOS SEQUENCIAIS

Circuitos Sequenciais Síncronos

- **ESTRUTURA GERAL**
  - Varáveis de entrada
  - Variáveis de saída
  - Variáveis de estado
  - Circ. combinatório
  - Memória



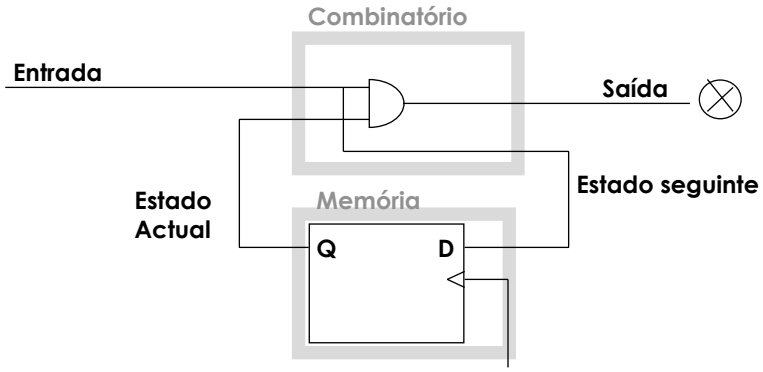
1

V.Lobo @ EN

CIRCUITOS SEQUENCIAIS

Circuitos Sequenciais Síncronos

- **Exemplo :**
  - Acender uma lâmpada quando a entrada se mantém durante 2 ciclos



2

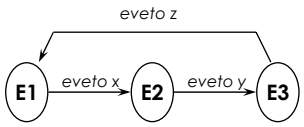
V.Lobo @ EN

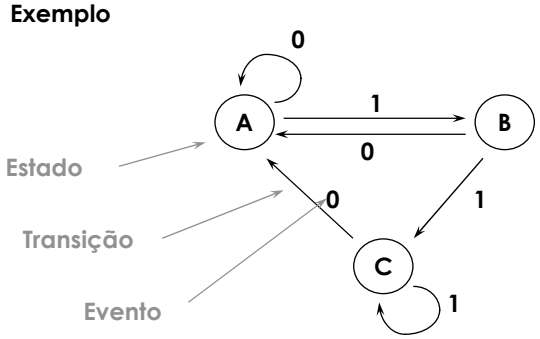
Nota: Estas folhas são apenas um guia de estudo e não contêm toda a matéria da cadeira

Modos de representar

Circuitos Sequenciais Síncronos

- **ESTADOS E DIAGRAMAS DE ESTADOS**
  - Um estado reflecte a história passada da máquina
  - Um diagrama representa os estados possíveis, e quais os eventos que levam a máquina de um estado a outro
  - Exemplo





A - Lâmpada apagada
B - Entrada activa há 1 ciclo
C - Lâmpada acesa

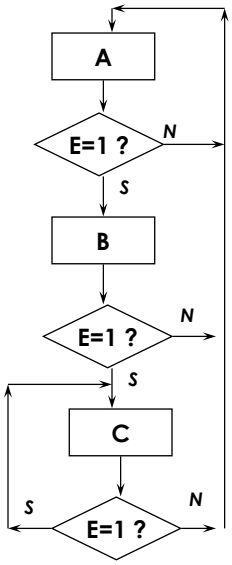
3

V.Lobo @ EN

Modos de representar

Circuitos Sequenciais Síncronos

- **Fluxogramas**
  - Semelhantes aos fluxogramas usados em software
  - Os estados são representados por blocos rectangulares (acções)
  - As transições são representadas por losângulos (decisões)
  - Exemplo:



4

V.Lobo @ EN

Tipos de circuitos sequenciais

- **MÁQUINAS DE MOORE**
  - **Saídas são função apenas do estado** (e não das entradas)
  - **Saídas variam sincronamente**
  - **São mais fáceis de desenhar mas podem necessitar de mais estados**
  - **No diagrama de estados, representam-se os valores das saídas juntamente com os estados**
- **MÁQUINAS DE MEALY**
  - **Saídas são função do estado anterior e das entradas**
  - **Saídas variam assincronamente com as entradas**
  - **São mais rápidas nas respostas e podem ter menos estados**
  - **No diagrama de estados, representam-se os valores das saídas juntamente com as transições de entrada num estado**

SÍNTESE DE SEQUENCIAIS SÍNCRONOS

- **1º Diagrama de estados**
  - **Passo mais criativo e mais crítico do processo**
- **2º Tabela de transições**
  - **Fazer uma tabela com todas as transições possíveis**
- **3º Eliminar estados redundantes**
  - **Objectivo: ter o número mínimo possível de estados**
  - **1º método - Inspecção da tabela de transições**
    - **→Se dois estados têm as mesmas saídas e os mesmos estados seguintes, então são o mesmo estado**
    - **→Não garante que todos os estados redundantes sejam eliminados**
  - **2º método - Método das partições**
    - **→Método sistemático para eliminar todos os estados redundantes**

SÍNTESE DE SEQUENCIAIS SÍNCRONOS

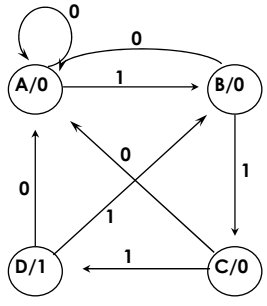
- **4º Codificação dos estados**
  - Atribuir a cada estado um código binário
- **5º Escolher flip-flops**
  - Os flip-flops JK são os mais usados pois permitem maior flexibilidade, e menos lógica externa
  - A síntese de circuitos é muito mais simples se se usarem flip-flops D (embora o circuito final seja mais dispendioso)
- **6º Obter as funções de excitação**
  - Dada a codificação dos estados (*n1* bits) e as entradas (*n2* bits) fazer um mapa de Karnaugh (de *n1+n2* entradas) para a entrada de cada um dos flip-flops de estado
- **7º Desenhar o logigrama do circuito e implementá-lo!**

V.Lobo @ EN

Exemplo

- **Pretende-se detectar a sequência 111 num conjunto de bits.**
  - 00100111010111011111010000111110100010
- **Diagrama de Estados**

Estado	Significado	Código
A	não recebi nada de interesse	00
B	Recebi um 1	01
C	Recebi dois 1	10
D	Recebi tres 1 e Encontrei !	11



V.Lobo @ EN

Exemplo

Circuitos Sequenciais Síncronos

- **Tabela de transições**

Estado	Entrada 0	Entrada 1
A/0	A	B
B/0	A	C
C/0	A	D
D/1	A	B

- **Eliminação de estados redundantes**
  - Não há, porque os estados A e D têm saídas diferentes
- **Funções de excitação para os Flip-flops e saída**
  - $D_0 = F(Q_0, Q_1, E)$ logo temos um mapa de Karnaugh a 3 variáveis
  - $D_1 = F(Q_0, Q_1, E)$

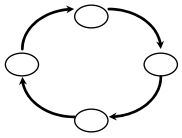
9

V.Lobo @ EN

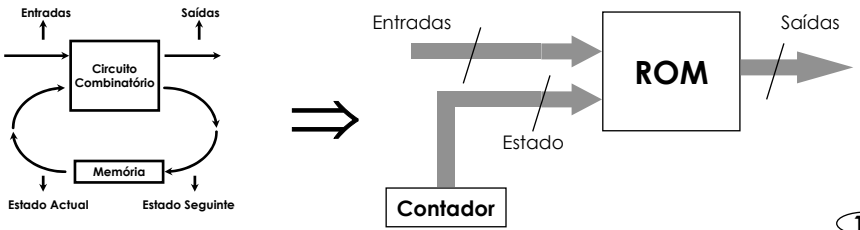
Controlo por ROM

Circuitos Sequenciais Síncronos

- **Em muitos problemas a sequencia de estados é fixa, formando uma** *cadeia linear*
  - →Ignição das velas num automóvel
  - →Robot de soldadura



  - A passagem ao estado seguinte pode ser feita com um CONTADOR
  - A parte combinatória do controlador pode ser implementada com uma ROM



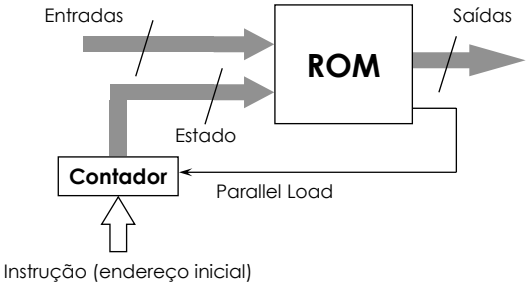
10

V.Lobo @ EN

Controlo por ROM

Circuitos Sequenciais Síncronos

- **Possibilidade de executar vários programas**
  - **O contador pode ser carregado com um valor inicial qualquer**
  - **Quando o "programa" chega ao fim, carrega um novo valor inicial para o contador**

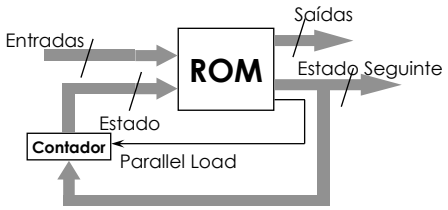


11

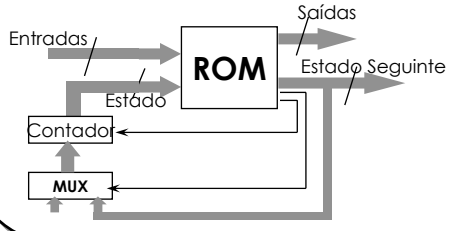
Controlo por ROM - Outras arquitecturas

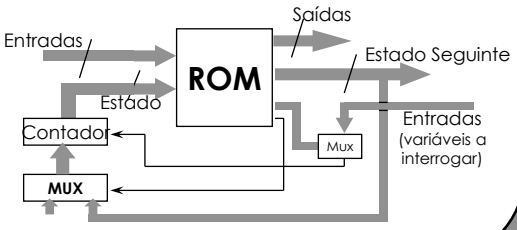
Circuitos Sequenciais Síncronos

- **Possibilidade de incluir saltos**
  - **Endereços seguintes possíveis guardados em ROM**
  - **Possibilidade de haver um ou mais endereços seguintes**



- **Máquinas de MOORE**
  - **Entradas só são usadas para gerar o parallel load e desmultiplexagem dos possíveis endereços seguintes**

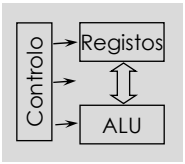




12

Microprocessadores

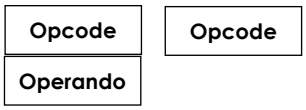
- **São sistemas que manipulam dados - máquinas de processamento**
  - **Podem fazer várias operações diferentes: cada uma delas é uma INSTRUÇÃO MÁQUINA**
  - **Uma sequência de operações (cada uma delas composta por vários passos) é um PROGRAMA**
- **Têm diversos componentes internos**
  - **ALU - Para fazerem operações de aritmética e lógica**
  - **REGISTOS - Para guardar dados durante o processamento**
  - **CONTROLO -Para controlar todo o sistema**



- **Executam programas que são sequências de instruções máquina**
  - **Ex.**
    - **→ADD A,B Soma o conteúdo do Registo A com o Registo B, guardando o resultado em A**
  - **Cada instrução máquina corresponde a um código numérico**
    - **→ADD A,B Tem o código 80H (128 em decimal)**

Instruções Máquina

- **Formato**
  - **Código de operação (OPCODE)**
  - **Operandos (Endereços ou dados)**



- **Operações típicas**
  - **Mover dados (entre registos ou memória)**
  - **Operações Elementares (AND,OR,ADD,etc)**
  - **Controlo de fluxo do programa (JUMP, CALL)**
  - **I/O (mover dados de/para periféricos)**

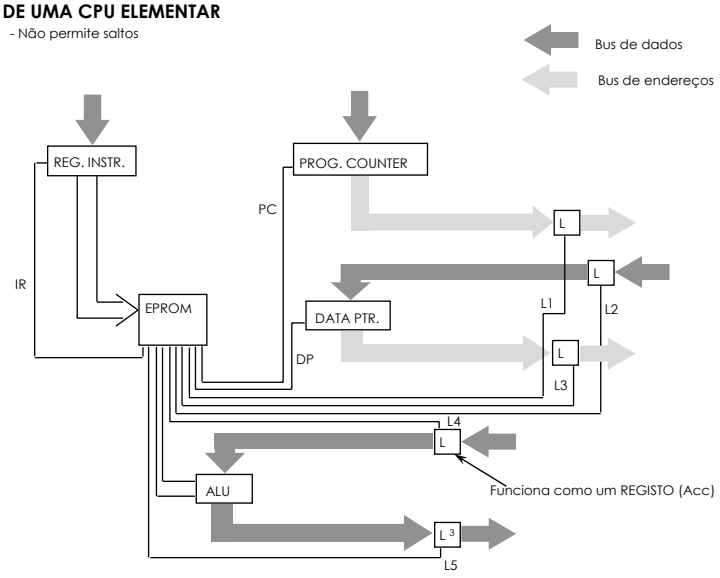
Exemplos:	
Instrução	Código
ADI 34	C6H, 22H
ADD B	80H
LXI B,2000	01H,00H,20H

- **RISC vs CISC**
  - **Conjunto grande de instruções => mais hardware**
  - **Conjunto pequeno de instruções => mais rápido**

MicroProgramação

EXEMPLO DE UMA CPU ELEMENTAR

- Não permite saltos



Microprogramação

- **Microprocessador como máquina a controlar**
  - **Conjunto de ALU, Registos, Buffers, etc**
  - **Uma instrução não é mais que uma sequência de passos**
- **Características do Microcódigo**
  - **Controlo directo dos sinais de comando do integrado**
  - **Permite a construção do "instruction set" que caracteriza o CPU ou microprocessador**
  - **Possibilidade do microcódigo ser reprogramável**
    - **→Emulação de outros computadores**
    - **→Normalmente o microcódigo é fixo (ROM)**
    - **→Novas versões por vezes têm o microcódigo re-escrito**

Implementação de Instr.Máquina

- **No CPU simples apresentado pretende-se uma instrução para complementar o conteúdo de um dado endereço**
  - **A mnemónica será: NOT *addr***
  - **Ex: NOT 20 deverá complementar o conteúdo do endereço de memória 20**
  - **A instrução deverá ter um byte para OPCODE e 1 byte de argumento para indicar o endereço de memória**
- **Sinais a controlar (total de 10 bits):**
  - **L1 …L5 controlo tri-state dos latches**
    - **→0 = alta impedância/keep**
    - **→1 = output enable/load**
  - **DP load enable dos registos**
    - **→ 0 = nil**
    - **→1 = Load**

Implementação de Instr.Máquina

- **Sinais a controlar (total de 10 bits):**
  - **PC Count enable do contador**
    - **→0 = nil**
    - **→1 = count**
  - **IR count/paralel load do contador**
    - **→0 = count**
    - **→1 = load**
  - **S0,S1 controlo da ALU**
    - **→00 - NIL Não faz nada**
    - **→01 - NOT Complementa o dado**
    - **→10 - INC Incrementa o dado**
    - **→11 - DEC Decrementa o dado**

Instruções Máquina

- **Passos da operação NOT *ADDR***
  - **1 - Ler endereço para DP**
  - **2 - Ler dado para o latch**
  - **3 - Fazer operação, guardar o dado**
  - **4 - Passar à inst. seguinte**

- **Observações**
  - **O 4º passo pode ser eliminado se no terceiro passo se fizer o pré-carregamento da instrução seguinte**
  - **Na prática, o Opcode faz o endereçamento de uma memória auxiliar que por seu turno irá fazer o endereçamento da ROM de controlo**
  - **Programação Horizontal vs Vertical**

L1 ……L5	DP	PC	IR	S0	S1
1 1 0 0 0	1	1	0	0	0
0 0 1 1 0	0	0	0	0	0
0 0 1 0 1	0	0	0	0	1
1 0 0 0 0	0	1	1	0	0

V.Lobo @ EN