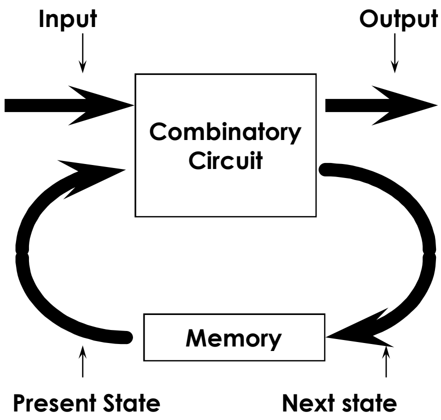
Hardware & Software for Information Technologies
Digital Controller Circuits

NovaIMS, V.1.1 V.Lobo 2018

Controllers and Sequential Circuits

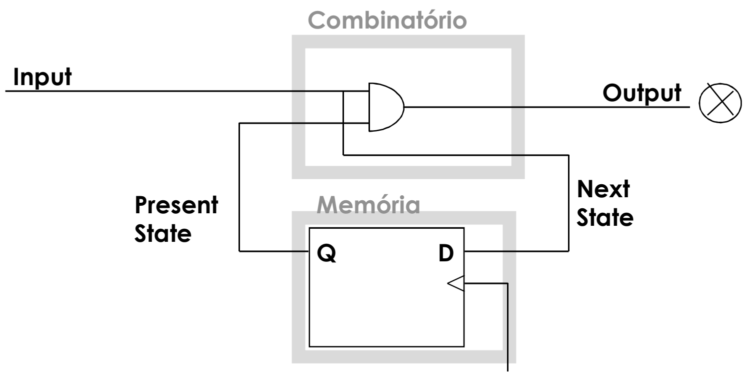
- **General Strucutre**
  - **INPUT** variables
  - **OUTPUT** variables
  - **STATE** variables

  - Combinatory Circuit
  - Memory



SEQUENTIAL CIRCUITS

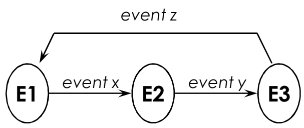
- **Example :**
  - Turn on a lamp if the input is ON for two consecutive clock cycles

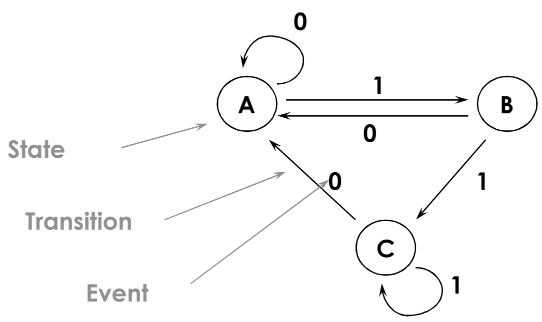


Nota: Estas folhas são apenas um guia de estudo e não contêm toda a matéria da cadeira

Ways to represente State Machines

- **States and State Machines**
  - A STATE reflects all the past history of the machine
  - A state machine DIAGRAM represents all possible states, and which events trigger the transition from one state do another
  - Exemple





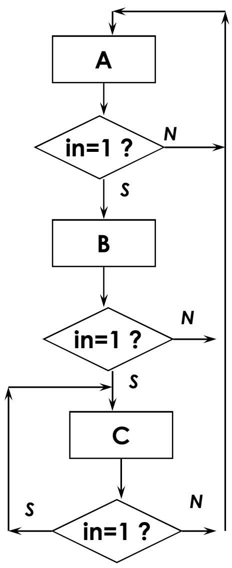
A – Initial state. Lamp is OFF

B – Input is active for less than 1 cycle, Lamp is still OFF

C – Lamp is ON

Ways to represente State Machines

- **Fluxograms**
  - Similar to software fluxograms
  - States are represented by rectangles (actions)
  - Transitions are represented by arrows and losangles (decisions)
  - Exemples:



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

Types of sequential machines

- **MOORE Machines**
  - **Outputs are function of only the STATE** (*i.e.* not the current inputs)
  - **Output change synchronously**
  - **They are easier to design, but may require more states**
  - **In a state diagram, the outputs are defined in each state**
- **MEALY Machines**
  - **Outputs are functions of the current state AND current inputs**
  - **Outputs may change asynchronously if inputs change.**
  - **They are faster in responding and require less states**
  - **In a state diagram, the outputs are defined for each transition**

Recipe for synthesising Sequential Circuits

- **1st Obtain the State Diagram**
  - **Most creative step of the whole process**
- **2nd Obtain the transition table**
  - **A table with all possible transitions**
- **3rd Eliminate reduntant states**
  - **Objective: have as few states as possible**
  - **1st method – Inspect the transition tables**
    - →**If two states have the same outputs and the same "next states", they are the same**
    - →**Dos not guarantee the all redundante states are eliminated**
  - **2nd method – Partition method**
    - →**Systematic way of eliminating all redundant states**

Recipe for synthesising Sequential Circuits

Circuitos Sequenciais Síncronos

- 4º Codify the states
  - Assign a binary code to each state
- 5º Choose the type of flip-flops
  - JK flip-flops are the most used because of their flexibility, and because they require little external logic
  - Synthesis is easier to follow when using D type flip-flops
- 6º Obtain the activation function
  - Given the codes of the presente state (*n1* bits) and the inputs (*n2* bits) make karnaugh maps (with *n1+n2* inputs) for the inputs to each of the state flip-flips
- 7º Draw the schema of the circuit and build it !

V.Lobo @ EN

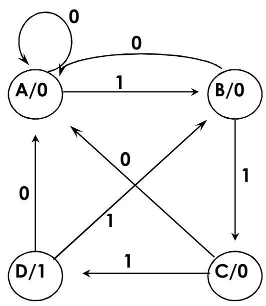
7

Example

Circuitos Sequenciais Síncronos

- Detect a sequence of thre 1's (111) in a stream of bits.
  - 0010011101011101111010000111110100010
- State diagram

State	Meaning	Code
A	nothing interesting	00
B	Received the first 1	01
C	Received the second 1	10
D	Received the third 1 Bingo !	11



V.Lobo @ EN

8

Example

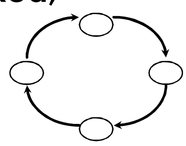
- **Transition table**

State	Input 0	Input 1
A/0	A	B
B/0	A	C
C/0	A	D
D/1	A	B

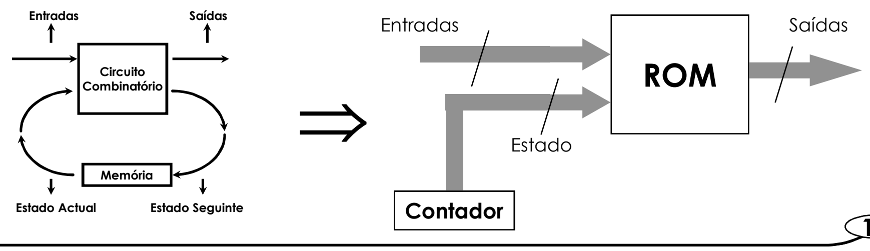
- **Remove redundant states**
  - There are none, since A and D have diferente outputs
- **Activation functions for the flip-flops and outputs**
  - $D_0 = F(Q_0, Q_1, E)$ thus, we have a 3 input Karnaugh map
  - $D_1 = F(Q_0, Q_1, E)$

Control circuits using ROMs

- **In many problems, the sequence of states is fixed, forming a *circular diagram***
  - →Sparkplug activation in a 4 stroke engine
  - →Soldering robot



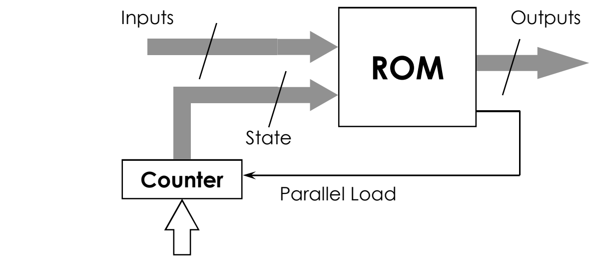
  - The change to the next state can be done with a COUNTER
  - The combinatory part can be implemented with a ROM



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

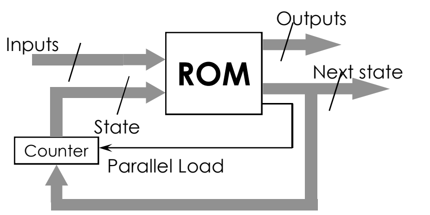
Control circuits using ROMs

- **Possibility of executing various programs**
  - **The counter may be loaded with different initial values (thus starting a different "program")**
  - **When a "program" reaches the end, a new initial value may be loaded into the counter**

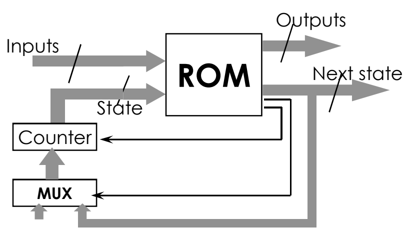


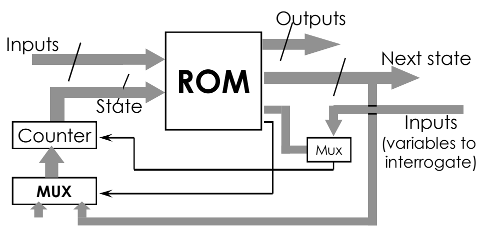
Control circuits using ROMs

- **Possibility of including JUMPS**
  - **Possible next addresses stored in ROM**
  - **Possibility of having more than one "next address"**



- **MOORE machines**
  - **Inputs are only used to generate the parallel load and demultiplex possible "next addresses"**

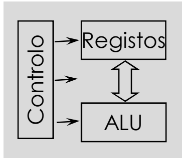




Microprocessors

Circuitos Sequenciais Síncronos

- **Control systems that process data**
  - Thay may perform severa, diferente operations, Each one is a MACHINE INSTRUCTION
  - A sequence of MACHINE INSTRUCTIONS is a PROGRAM
- **Internal componens of a microprocessor**
  - ALU - To perform Arithmetic and Logic operations
  - REGISTERS - To store data
  - CONTROLO – To control the whole sysytem



- **They execurte programs (that are sequences of MACHINE INSTRUCTIUONS)**
  - Example
    - →ADD A,B Adds the contents of register A with the contentes of register B, and stores the result in register A
  - Each machine instruction corresponds to a numerical code
    - →ADD A,B Is code 80H (128 in decimal)

V.Lobo @ EN

13

Machine instructions

Circuitos Sequenciais Síncronos

- **Format**
  - Operation Code (OPCODE)
  - Operands (addresses or data)

Opcode	Opcode
Operand	

- **Typical operations**
  - Move data (between registers or memory)
  - Elementary operations AND,OR,ADD,etc)
  - Flux control within the program (JUMP, CALL)
  - I/O (move data to/from peripherals)

Exemples:

Instruction	Code
ADI 34	C6H, 22H
ADD B	80H
LXI B,2000	01H,00H,20H

- **RISC vs CISC**
  - Reduced Instruction Set Computer
    - →Few instructions, each instruction is fast, less hardware
  - Complex Instruction Set Computer
    - →Specialized instructions, more hardware

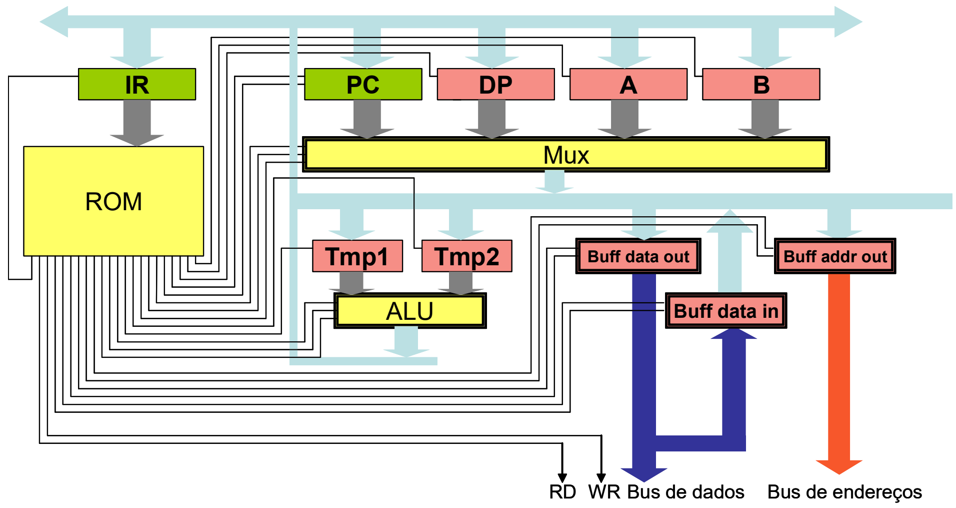
V.Lobo @ EN

14

MicroPrograming

Circuitos Sequenciais Síncronos

EXEMPLE OF na ELEMENTARY CPU (DAE02)



15

Meaning of each line

Circuitos Sequenciais Síncronos

1 – IR: 1 ⇒ provoca um PARALLEL LOAD no contador IR; 0 ⇒ deixa que o contador IR passe ao número seguinte.
2 – RD: Sinal de controlo enviado para o BUS. 1 ⇒ provoca uma operação de leitura à memória.
3 – WR: Sinal de controlo enviado para o BUS: 1 ⇒ provoca uma operação de escrita à memória.
4 - "Buffer data in" output enable: 1⇒ o registo buff_data_in põe o seu conteúdo à saida. 0 ⇒ as saídas ficam em tri-state.
5 - "Buffer data in" load :1 ⇒ o registo buff_data_in carrega o que tiver à entrada. 0 ⇒ as mantém o seu estado.
6 - "Buffer data out" output enable: 1 ⇒ o registo buff_data_out põe o seu conteúdo à saida. 0 ⇒ as saídas ficam em tri-state.
7 - "Buffer data out" load: 1 ⇒ o registo buff_data_out carrega o que tiver à entrada. 0 ⇒ as mantém o seu estado.
8 - "Buffer addr out" output enable: 1 ⇒ o registo buff_addr_out põe o seu conteúdo à saida. 0 ⇒ as saídas ficam em tri-state.
9 - "Buffer addr out" load: 1 ⇒ o registo buff_addr_out carrega o que tiver à entrada. 0 ⇒ as mantém o seu estado.
10- ALU output enable- Quando é 1, a ALU põe o seu conteúdo à saida. Quando é 0, as saídas ficam em tri-state.
11- ALU S_0 - Sinal que em conjunto com ALU S_1 contola a opração efectuada pela ALU. Com S_1S_0 = "00" a ALU faz uma soma de TMP1 com TMP2, com "01" faz TMP1-TMP2, com "10" faz a operação AND entre os operandos, e com 11 faz OR.
12- ALU S_1 - Outro sinal de controlo da ALU-
13- TMP1 load - Carrega o registo TMP1.
14- TMP2 load - Carrega o registo TMP2.
15- MUX output enable - Quando é 1, o MUX põe um dado à saída. Quando é 0, as saídas ficam em tri-state.
16- MUX S_0 - Sinal que em conjunto com MUX S_1 faz a selecção do que o MUX põe à saída. S_1S_0="00" selecciona PC, "01" selecciona DP, "10" selecciona A, e "11" selecciona B.
17- MUX S_1 - Outro sinal de controlo do MUX.
18 – PC load - Faz um PARALLEL LOAD ao contador PC.
19- PC increment - Incrementa o contador PC.
20- DP load - Carrega o registo DP. / 21- A load - Carrega o registo A 722- B load - Carrega o registo B.

16

Microprograming

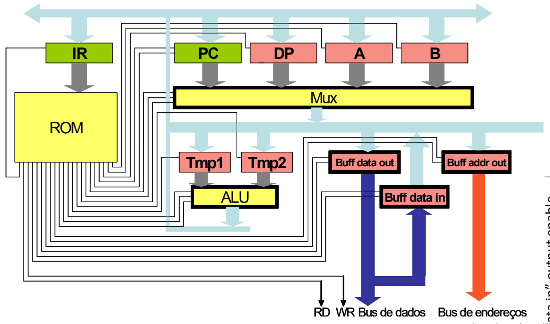
- A Microprocessor is a machine that we need to control
  - Set of ALU, Registers, Buffers, etc
  - A MACHINE INSTRUCTION is no more than a sequence of steps (activation/deactivaton of control lines)
- Characteristics of microcode
  - Direct control of the inputs to each component
  - The set of all microprograms consitutues the "instruction set" that caracterizes the CPU or microprocessor
  - Possibility of reprogramming the microcode
    - →Emulation of other computers
    - →Normally the microcode is fixed (in ROM)
    - →Updates to existing micoprocessors may have the microcode re-written

Machine Code

- Steps of the ADD A,B operation:
  - 1 – Put the contents of A in TMP1
  - 2 - Put the contents of B in TMP2
  - 3 – Add the data in the ALU, store the result in A
  - 4 – Get the next instruction
- Comments
  - The 4th step can be avoided if the 3rd stepe pre-loads the next instruction
  - In practice, the OPCODE addresses an auxiliary memory, that in turn addresses the control ROM
  - Horizontal vs Vertical micro-programming

Exemples

Circuitos Sequenciais Síncronos



Signals to be controlled (total of 22 bits):

	IR	RD	WR	"Buffer data in" output enable	"Buffer data in" load	"Buffer data out" output enable	"Buffer data out" load	"Buffer addr out" output enable	"Buffer addr out" load	ALU output enable	ALU S0	ALU S1	TMP1 load	TMP2 load	MUX output enable	MUX S0	MUX S1	PC load	PC increment	DP load	A load	B load
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1																						
2																						
3																						
4																						
5																						
6																						
7																						
8																						

Implement the following instructions:

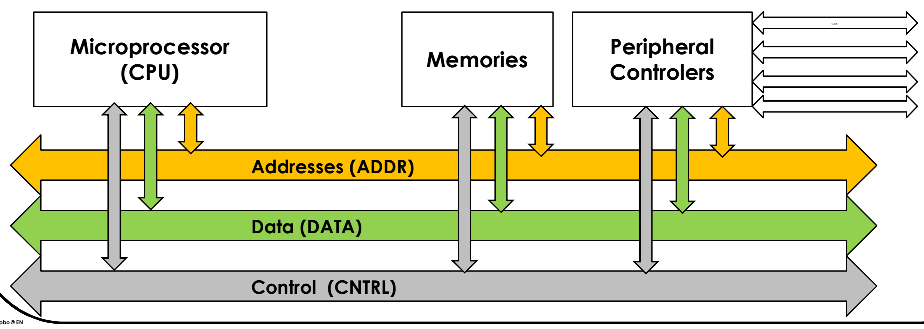
- Copy to B what is in A
- Add A and B, leaving the result in A
- Copy to A what is in the memory address pointed to by DP

19

Microprocessor versus peripherals

Circuitos Sequenciais Síncronos

- The microprocessor is conneted to the other componentes through a SYSTEM BUS, that can be devided into:
  - ADDRESS BUS - Indicates WHERE you want to read/write
  - DATA BUS - Indicates WHAT you want to read/write
  - CONTROL BUS - Provides the necessary control and synchronization signals to perform the operation (indicates whether you want to read or write, in memory or contreolers, etc)



20