The Project of a Didatic CPU for Multidisciplinarity

FERLIN, Edson Pedro

Pontificia Universidade Catolica do Parana, Laboratorio de Logica Programavel - Eng. Computacao, Caixa Postal 6432 Cep. 80.020-010, Curitiba-Parana-Brasil, ferlin@ccet.pucpr.br

Abstract: *An ongoing trend in teaching is to employ multidisciplinarity to help students understand concepts involved in more than one subject, contextualizing the knowledge, and avoiding its segmentation. In this way, the students enrolled in Digital Systems, in the Computer Engineering program, have as project activity the elaboration of a microprogramed CPU, which, besides tackling concepts related to digital systems, allows the student to learn about Microprocessors, making available an environment of discovery and exploration, and contributing to a better perception of the possibilities of the area.*

Keywords: *Digital Systems, Microprocessors, CPU, Multidiciplinarity.*

1 Introduction

An ongoing trend in teaching is to employ Multidisciplinarity, contextualizing the knowledge, and avoiding its segmentation.

In this way, the students enrolled in Digital Systems, in the Computer Engineering program, have as project activity the elaboration of a 4-bit microprogramed CPU (*Central Processing Unit*) [CHR-96], [LIM-98], [TAN-92], also called processor, a device which is the brain of computer systems, being responsible for processing information, and for controlling other components, internal or external.

This project joins the concepts of Digital Systems: Digital Logic, Combinational Circuits, Sequential Circuits [HIL-81] and Memories in Semiconductors: ROM (*Read Only Memory*) and RAM (*Random Access Memory*), together with the basic concepts of Microprocessors: data flow, program memory, *opcode*, instructions, programs, operands, microinstructions and others [CHR-96], [LIM-98], [TAN-92].

This work is carried out at the end of the sixth semester in the course of Digital Systems and it serves as basis for the Microprocessors course (offered in the following semester), allowing the students to join the concepts of one course with those coming from another, giving sequence to the content, and avoiding its segmentation.

The project meets to the educational need of allowing the student to develop so much the theoretical as practical concepts, already approached in the course of Digital Systems, with others that will still be explored in detail in the Microprocessors course.

The students develop this project after they have seen, in the theory and practice, all the involved components, as well as projected some of them, allowing that they use this knowledge in its work, joining the information obtained previously in the Digital Systems course with the others that will be approached in sequence of the project and that are topics in the course of Microprocessors.

The specific concepts of microprocessors will be reviewed in detail in the Microprocessors course, whose pedagogic revenue will be superior, by virtue of they have already used in the practice the information of this domain and that were sediment with this project.

2 The Content of the Courses

For best to understand the purpose of this project, it is necessary to know the contents approached in each one of the course that are directly involved in the work.

In the course of Digital Systems, offer in the fifth and in the sixth semester, the content is divided in two basic parts: Combinational circuits and Sequential circuits and, as advanced topics, the Memories in Semiconductors and the Programmable Devices [FER-98]: PLA (*Programmable Logic Array*) and PLD (*Programmable Logic Device*).

In the relative part to the Combinational circuits have the truth function, logical gate, boolean algebra, algebraic reduction, combinational logic circuits, *Karnaugh*-maps reduction, *Quine-McCluskey* reduction method and basic logical circuits as for example: ALU (*Arithmetic - Logic Unit*), decoders, multiplexes, bus and others [HIL-81].

With respect to sequential circuits the subjects covered are flip-flops, counters, registers, synchronous and asynchronous sequential machines, synthesis of sequential circuits, projects of sequential machines, synthesis of state diagram (state machine) [HIL-81].

And in the topic on Memories in Semiconductors, one covers the Memories RAMs and ROMs, the types, structures, block diagram, principle of operation, basic cell [CHR-96] , and functions and application of programmable devices [FER-98].

The Microprocessors course, offered in the seventh and eighth periods, covers the banks of memory, control and operation of the processors, microprogramming [TAN-92], instructions, programs, *assembly* language programming, and project in particular with microprocessors with Microcontroller 8051 of Intel.

3 The Project

This project is based on the contents studied in the Digital Systems course, but with some topics seen in Microprocessors, allowing the use of the multidisciplinarity, as way to integrate these courses.

In this project the following logical components are used, which were studied in Digital Systems:

- Logic gate;
- ALU (*Arithmetic – Logic Unit*);
- Bus;
- *Tri-State* devices;
- Registers;
- Counters;
- Memories: ROMs and RAMs.

These components, when seen separately, they don't make much sense, but linked they generate components that have specific functions, as it is the case of the data flow of a CPU, that is composed by ALU, registers, *tri-state* devices and bus [CHR-96] .

In the Microprocessors course the components that in Digital Systems were basic and had generic functions, as is the counters' case, now have a specific function, in the program counter's case, that indicates which instruction to be executed next, and of microprogram, which points to the next microinstruction that will be activated.

The same principle is valid for the ROM Memory that have the function of storing information (data), for a period of time relatively large, without the device energized and, in this case, has the function of controlling memory, storing the microprogram or specifically the microinstructions, and operate as a combinational circuit, because any alteration in its inputs (addresses), it almost produces that immediately, due to intrinsic characteristic of the component, the alteration of the data in the output.

3.1 Description of the Project

The project is based on the development, with didactic aspect, of a CPU of 4 bits (see Figure 1), with a minimum set of instructions.

It contains three different functional blocks: Data Flow – that is the heart of CPU, Control Memory - responsible for the control (activation) of all the components and, last, but not less important, the Program Memory, that has as function to store the program that will be executed by the CPU.

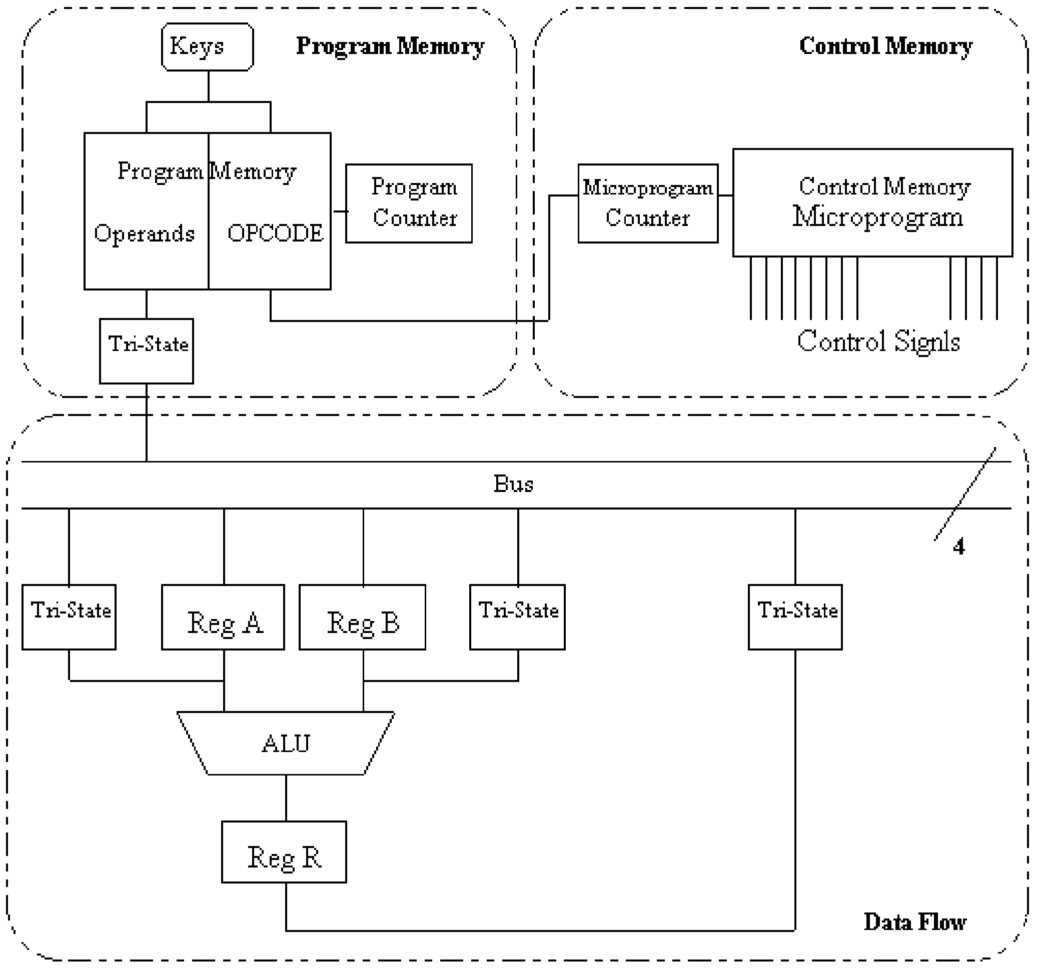

Figure 1 - Diagram in Block of CPU

The Data Flow block, composed of four bits, which is the heart of CPUs, is composed by a ALU - responsible for the arithmetic and logical operations, three registers (reg), storage elements, that are denominated A, B and R, and four *tri-state* devices for the isolation of the bus, which serves as connection for the other elements.

The Control Memory block is composed of a preset counter, called microprogram counter, that points to the microprogram, and to at least, an EPROM (*Erase Programmable Read Only Memory*), which stores the microprogram or, more specifically, the microinstructions.

It is from this block that originate the signals that will control all the other components, that is, the control lines for the components of the system, so much internal to CPU as well as external, coming from the output of the EPROM memory.

The microprogram which contains microinstructions, is stored in control memory, and each one of these microinstructions is in a position of the memory, in the microoperations form.

To each microinstructions sequence corresponds an instruction of CPU, and the set of these instructions is called set instructions.

The content of this memory is fixed, because it contains the set of instructions of CPU or, more specifically, the several microinstruction sequences that correspond to each one of the valid instructions for this CPU and that cannot be altered during the execution.

The Program Memory block is composed of two Memories of the type RAM, one to store the *Opcode* of the instruction and another for operand (if the instruction contains that), because there are instructions such as NOP (*No Operation*) that don't need an operand; and for a program counter, that indicates which instruction that will be executed next by the CPU.

In this block the program, sequence of instructions in a logical order, is stored, that the user defines using the instructions of CPU, by means of the input keys and in each position of the program memory it is an instruction that is divided in two parts: one for *Opcode*, that determines the action to be taken and another for Operand, data on which the action is taken.

The data of *Opcode* serves to initialize the microinstructions counter, which will point to the first of a set of necessary microinstructions so that determine the action and operand of the instructions will go directly to the bus of the data flow that will be stored in one of the internal registers, for later processing.

The content of the program memory is variable, because it will contain the sequence of instructions defined by the user in a certain order, that is, the program application that the user has developed to a computational demand.

Based on this description and on the block diagram of the CPU (Figure 1), the project is elaborated, comprising the implementation and the documentation of a 4-bit microprogramed CPU, that should execute at least the following instructions:

✍	MOV	A, <*src*>	; A ? ?? ? ? ? ? ?
✍	MOV	B, <*src*>	; B ? ?? ? ? ? ? ?
✍	ADD	A, B	; A ? ?? ?? ??
✍	INC	A	; A ? ?? ?? ??
✍	OR	A, B	; A ? ?? ?? ? ??

The sequence of these instructions, in a logical order informed by the user, defines the program that will be executed by the CPU, by means of the actions given by the microinstructions that compose each one of the instructions used in the program.

3.2 Operation

Initially the program counter points to the position decimal 0 (0000 in binary) of the program memory, where it will be the first instruction of the program (in binary code). This code (*opcode*) will be the initial data of load for the microprogram counter, that is, it will indicate which is the first microinstruction of the sequence, that corresponds to that instruction requested by the user, that will be executed.

At each *clock* pulse, the microprogram counter will point to the next microinstruction, in the control memory, to be executed. When all the microinstruction sequences have been executed, the program counter is automatically increased by the control memory by one unit, so that it points to the next instruction of the program, that is in the program memory. Starting from this point, the cycle repeats until the program is fully executed.

As it can be observed in Figure 1, the connection among the different blocks occurs by means of the buses.

In the case of the connection among the block of the program memory and of control, this occurs through the bus that connect the memory, where the *opcode is* stored, to the microprogram counter and that is extremely necessary, because the data of the *opcode* presets the counter indicating the first microinstruction to be executed.

Between the block of the program memory and the data flow, we have the bus of data internal to the CPU, which maintains the data flow isolated from the program memory (operands) for a *tri-state* device. This connection allows the operand to go to the data flow to be processed.

From the control memory emanate the control lines that are the responsible for controlling each one of the components of the functional units that compose CPU.

3.3 Example of CPU

Based on the description of the project, assuming that, for example, CPU implements the following instructions (Figure 2):

Instruction	***Opcode***
MOV A, data	***0000***
MOV B, data	***0011***
MOV B, A	***0101***
ADD A, B	***1000***
INC A	***1010***
OR A, data	***1110***

Figure 2 - Instructions and Opcode of a CPU example

Then, in the control memory (microprogram), that is in EPROMs, we will have the following content, leaving the presupposition that we are using four bits to specify the position in memory (Figure 3):

Position	**Microinstruction**	**Comment**
0000	**11110001010101... .111**	**Microinstructions of the**
0001	**11010101010111... .011**	**instruction MOV A, data**
0010	**10000011010101... .000**	
0011	**....**	**Microistructions of the**
0100	**....**	**instruction MOV B, data**
0101	**....**	**Micro instructions of the**
0110	**....**	**instruction MOV B, A**
0111	**....**	
1000	**....**	**Micro instructions of the**
1001	**....**	**instruction ADD A, B**
1010	**....**	**Micro instructions**
1011	**....**	**of the instruction INC A**

1100	**....**	
1101	**....**	
1110	**....**	**Micro instructions of the**
1111	**....**	**instruction OR A, data**

Figure 3 - Control memory (microprogram) of the CPU example

Considering that the program that the user developed, for example, be the following:

```
MOV    A, 1000     ; A ?  ??????
MOV    B, 0010     ; B ?  ?????
ADD    A, B        ; A ?  ?? ?? ??
MOV    B, A        ; B ?  ??
INC    A           ; A ?  ?? ?? ??
OR     A, 0110     ; A ?  ?? ?? ? ??????
```

Then, in the program memory (RAMs) we will have the following content (Figure 4) considering the sixteen positions of the memory, reminding that the position of the memory is determined physically, for the address of its cells of memory, where the *opcode is* in a RAM, operand in other, and the field comment doesn't exist physically in the memory, and is only used for remarks about the program.

Position	***Opcode***	**Operands**	**Comment**
0000	***0000***	***1000***	***MOV A, 1000***
0001	***0011***	***0010***	***MOV B, 0010***
0010	***1000***	***xxxx***	***ADD A, B***
0011	***0101***	***xxxx***	***MOV B, A***
0100	***1010***	***xxxx***	***INC A***
0101	***1110***	***0110***	***OR A, 0110***
0110			
0111			
1000			
1001			
1010			
1011			
1100			
1101			
1110			
1111			

Figure 4 - Program memory of the CPU example

Obs: The **xxxx** indicates that the data, in this case, can be neglected.

4 Conclusion

Based on this idea, the student can suggest alterations and improvements that make part of the creativity and initiative of each team, creating constructive approach.

In this work, besides the concepts related to Digital System, the student can assimilate some knowledge of Microprocessors, such as microprogram, microinstructions, data flow, programs, instructions, *opcode* and operand, and others, creating an environment of discovery and exploration, contributing to a better perception of the possibilities of the area.

In this way, it was possible to create a connection among the courses of the digital area, in the Computer Engineering program, in this case Digital Systems and Microprocessors, allowing the student to build a solid relationship among the contents studied in different courses.

Topics that a lot of times stagnate the courses, now are part of universe that is integrated, facilitating a better understanding and interrelations.

The project is evaluated in the practice, in laboratory, to check its operation, and at the end of the work the students should write a report, according to the technical norms, which is constituted of several items, as for example specifications, project, components, adopted procedures, operation, results, conclusions, consulted bibliography, and others and that comes to consolidate the learning.

The report of the project can be given printed so much in paper, as well as (for recommendation) in storage media, as in diskettes or CD-ROM, using multimidia resources of *hiperlink*, that besides facilitating the browsing of the report, also turns it more pleasant and friendly, which could also be available in data base, facilitating the access to the works, for example for the *internet*, and serving as reference for future work.

This fact meets another extremely important need, which is dissemination of the knowledge, the acquired information, allowing it to be used by other students and also by the community in a general.

References

[CHR-96] - Christiansen, Donald, Editor, "Digital Computer Systems", *Electronics Engineer's Handbook*, Fourth Edition, New York-USA: IEEE Press, 1996, pp. 27.1-27-94.

[FER-98] - Ferlin, E.P., Eleuterio, M.A.M., "Vantagens na Elaboracao de Projetos Digitais Utilizando Dispositivos Logicos Programaveis", *IV Simposio de Pesquisa e Extensao em Tecnologia*, Natal-Brazil: UFRN, 1998, pp. 97-99.

[HIL-81] - Hill, F.J., Peterson, G.R., Introduction to Switching Theory and Logical Design, Third Edition, New York-USA: John Wiley & Sons, 1981.

[LIM-98] - Lima de, Jose A. G. *et al*, "A Flexible Microprogrammable Controler for ATM Swith", *ICMP-98 International Conference on Microelectronics and Packaging*, Proceeding I, Curitiba-Brazil: LAC/LACTRO, 1998, pp. 221-228.

[TAN-92] - Tanenbaum, Andrew S., "O Nivel de Microprogramacao", *Organizacao Estruturada de Computadores*, Rio de Janeiro-Brazil: Editora Prentice-Hall, 1992, pp. 128-181.