# HANDS-ON PROCESSOR DESIGN EXPERIENCE FOR COMPUTER ORGANIZATION AND ARCHITECTURE STUDENTS

Hala ElAarag
Department of Mathematics and Computer Science
Stetson University
421 N. Woodland Boulevard,
DeLand, FL 32723

## Abstract

Computer organization and architecture are core courses in the computing curricula. Providing a good applied experience in these courses is crucial for all computing disciplines. In this paper we present the design of an instruction set architecture to address the need of providing a simple but realistic hands-on experience on the hardware level to computer organization and architecture students. The control unit is the most tedious part of any processor design. Our goal is to show students how they can build a complete pipelined processor in the lab with minimum cost using TTL chips. In this paper we continue our previous work and show two approaches to the design of the control unit. We hope that with this effort we provide students with a deep insight on processor design that cannot be provided with theory or simulation.

## Introduction

It is very important to combine theory and practice when it comes to teaching computer organization. There are many simplified architectures designed for pedagogical purposes. The Essentials of Computer Organization and Architecture by Null [1] introduces assembly language through MARIE. MARIE has a very simplistic instruction set and datapath. Hennessy and Patterson's Computer Organization and design [2] use MIPS as the example for assembly language programming. Some instruction set architectures are designed with the goal of prototyping using FPGAs [3] while others are based on Hardware description Languages (HDL) [4, 5]. Though, most of them are designed as simulators, some examples are Ant32 [6] and MARS [7]. Yunten Labs [8] provide an architectural kit called the Computer Architecture Lab (CAL) to help students build a simple processor. However, this system is a very simple 4-bit architecture with a handful of instructions.

Our goal was to design an instruction set architecture that provide a rich experience and a clear understanding of the details of instruction execution at the hardware level. Students can build their own processors from scratch and follow the different stages of instruction execution starting from the fetching, decoding, executing and writing back to memory.

DLX is a simplified Reduced Instruction Set Architecture (RISC) architecture that is designed mainly for pedagogical purposes. In [9], we took DLX one step further, and reduced its instruction set based on the Standard Performance Evaluation Corporation (SPEC) benchmarks [2] to reach our proposed (HRISC) instruction set. We presented the table of events of HRISC that is considered as the documentation that shows the steps of execution of the different HRISC instructions in the pipeline. We then presented a complete design of HRISC and showed how to build it in a lab using off the shelf TTL chips. The control unit is the most difficult part in processor design. In this paper, we provide the details of the design of HRISC control unit using two approaches. The first is a microcoded control unit while the other is a hardwired control unit.

## HRISC Instruction architecture

HRISC has two instruction formats: an immediate type and a register to register type. They are shown in Figure 1. There are 17 instructions in HRISC instruction set as shown in Table 1.

Figure 1: HRISC Instruction Format.

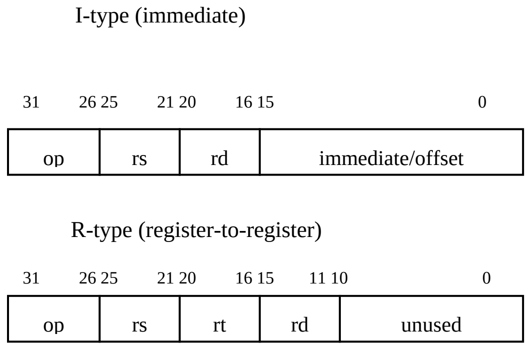


Table 1: HRISC Instruction Set.

| **Control** | |
|---|---|
| Branch | BEQZ, BNEZ |
| Jump | JR , JALR |
| **Arithmetic/logical** | |
| Add | ADD,ADDI |
| Subtract | SUB |
| And Logical | AND |
| Exclusive Or | XOR |
| Shift Right Logical | SRL |
| Shift Left Logical | SLL |
| Load High Immediate | LHI |
| Set | SEQ, SLT, SGT |
| **Data Transfer** | |
| Load | LW |
| Store | SW |

## Microcoded Control Unit

HRISC can be divided into five basic execution steps, where each step becomes a pipe stage:

1. IF - instruction fetch
2. ID - instruction decode and register fetch
3. EX - execution and effective address calculation
4. MEM - memory access
5. WB - write back

Figure 2 shows the block diagram of the pipelined HRISC. It is divided into the five stages to show clearly the flow of instructions in the pipeline. Every stage contains its microprogram PROM which contains the corresponding microinstructions. These microinstructions output the control lines used to control the flow in this stage. Operations performed in each stage are drawn horizontally on the same level to indicate that they are actually performed at the same time. For example IR1 and PC1 are at the same level in the IF stage. In the ID stage IR2, A, B, and PC2 are at the same level. The same thing applies for IR3, OUT1, COND, MAR and MDR1 in the EX stage. A more detailed block diagram can be found in [9].

This section shows the microinstruction format of each PROM, and the possible entries for each field.

**PROM ID**

| | | |
|---|---|---|
| • b1,b2 | 10 | BEQZ |
| | 11 | BNEQZ |
| | 01 | JR, JALR |
| | 00 | otherwise |

**PROM EX**

| | | |
|---|---|---|
| • EB | 0 | ADD, SUB, AND, XOR, SHIFT, SET |
| | 1 | ADDI, LOAD, STORE, LHI |
| | x | O.W. |
| • New | 1 | LHI |
| | 0 | O.W. |
| • SETF | 0 | SEQ |
| | 1 | SLT, SGT |
| | x | O.W. |
| • SR | 1 | SRL |
| | 0 | O.W. |
| • SL | 1 | SLL |
| | 0 | O.W. |
| • S3-S0 | 0001 | ADD, ADDI, LHI |
| | 0110 | SUB,SEQ,SLT, SGT |
| | 1011 | AND |
| | 0110 | XOR |
| | x | O.W. |
| • Cn | 0 | SLT |
| | 1 | ADD, ADDI, LHI, SUB, SEQ; SGT |

x O.W.

- M 0 ADD, ADDI, LHI, SUB, SEQ, SLT, SGT
  1 AND, XOR
  x O.W.
- OUTI ← A op B 1 ADD, ADDI, LHI, SUB, AND, XOR
  0 O.W.
- OUT3 ← A op B 1 SET
  0 O.W.

**PROM MEM**

- OUT2 ← OUT 1 1 ALU
  0 O.W.
- OUT2←OUT3 1 SET
  0 O.W.
- MDR2← DM [MAR] 1 LOAD
  0 O.W.
- DM [MAR] ← MDR1 1 STORE
  0 O.W.

**PROM WB**

- RD ← OUT2 1 ALU, SET
  0 O.W.
- R31←LINK3 1 JALR
  0 O.W.
- RD←MDR2 1 LOAD
  0 O.W.

## Hardwired Control Unit

Since the PROMS happen to have most of their entries as don't cares, also they have very few control lines, therefore it is better to hardwire the control unit. This would lead to less cost and better speed. The following section describes how the HRISC control unit could be hardwired.

Each instruction has a 6-bit opcode, they are named a,b,c,d,e and f. To facilitate the job, the instruction's opcode can be assigned values such that each bit of the opcode decides which group the instruction belongs to. That is, bit "a" decides whether the instruction is R-type or I-type, "b" decides whether it is a memory or a control instruction, bits "c" and "d" differentiate between different memory/control instructions, "e" decides whether it is a shift instruction from the ALU R-type instructions, while "f" differentiates between ALU and SET instructions, as follows:

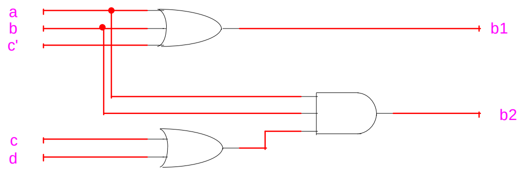
| opcode | |
|---|---|
| a | 0 R-type<br>1 I-type |
| b | 0 memory<br>1 control |
| e | 0 shift<br>1 non-shift R-type |
| f | 0 R-type ALU<br>1 R-type SET |

The following Table shows the opcode bit configuration of the instructions.

| | a | b | c | d | e | f |
|---|---|---|---|---|---|---|
| **LW** | 1 | 0 | 0 | 0 | X | X |
| **SW** | 1 | 0 | 0 | 1 | X | X |
| **LHI** | 1 | 0 | 1 | 0 | X | X |
| **ADDI** | 1 | 0 | 1 | 1 | X | X |
| **ADD** | 0 | X | X | X | 1 | 0 |
| **SUB** | 0 | X | X | X | 1 | 0 |
| **AND** | 0 | X | X | X | 1 | 0 |
| **XOR** | 0 | X | X | X | 1 | 0 |
| **SRL** | 0 | X | X | X | 0 | 0 |
| **SLL** | 0 | X | X | X | 0 | 0 |
| **SEQ** | 0 | X | X | X | 1 | 1 |
| **SLT** | 0 | X | X | X | 1 | 1 |
| **SGT** | 0 | X | X | X | 1 | 1 |
| **BEQZ** | 1 | 1 | 0 | 0 | X | X |
| **BNEQZ** | 1 | 1 | 0 | 1 | X | X |
| **JR** | 1 | 1 | 1 | 0 | X | X |
| **JALR** | 1 | 1 | 1 | 1 | X | X |

## Hardwiring PROM ID

PROM ID has two control lines b1 and b2, they can be hardwired as follows:



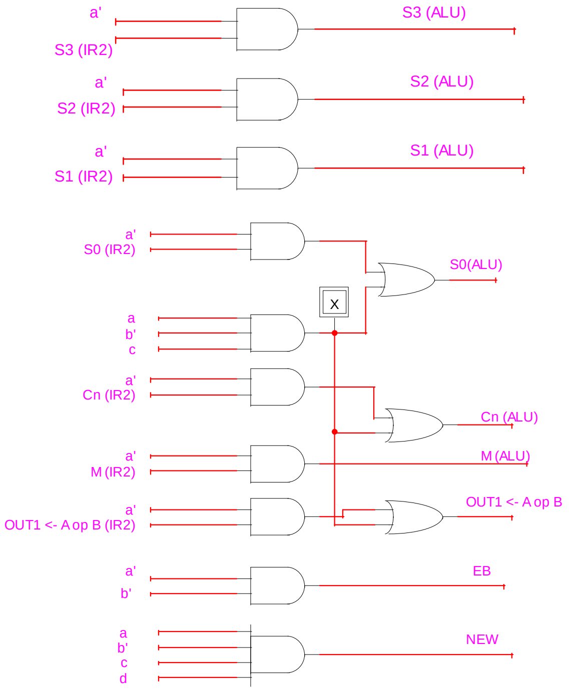
## Hardwiring PROM EX

PROM EX has 13 control lines. Since the R-type instruction format has 11 unused bits, then these bits could be used as control bits for the R-type instructions. The 11 control lines that are put in the R-type instruction are: S3-S0, Cn, M, OUT1 ← A OP B, OUT3 ← A OP B, SETF, SR, SL. Then the R- type instructions could be assigned the following values:

| | | | | | |
|---|---|---|---|---|---|
| ADD | 0xxx10 | rs | rt | rd | 00011010x00 |
| SUB | oxxx10 | rs | rt | rd | 01101010x00 |
| AND | 0xxx10 | rs | rt | rd | 1011x110x00 |
| XOR | 0xxx10 | rs | rt | rd | 0110x110x00 |
| SLR | 0xxx00 | rs | rt | rd | xxxxxx00x01 |
| SLL | 0xxx00 | rs | rt | rd | xxxxxx00x10 |
| SEQ | 0xxx11 | rs | rt | rd | 01101001000 |
| SLT | 0xxx11 | rs | rt | rd | 01100001100 |
| SGT | 0xxx11 | rs | rt | rd | 01101001100 |

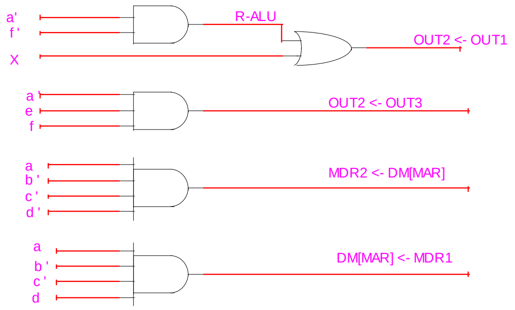
As for the I-type instructions a decoder is used to decode bits b, c and d of the opcode.

The PROM is hardwired as follows:



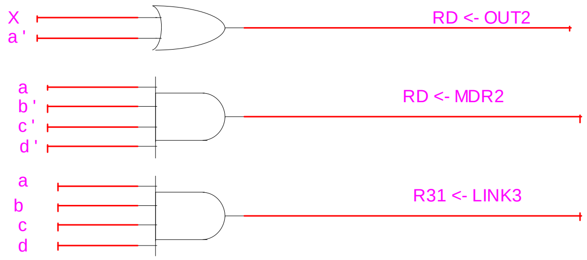
## Hardwiring PROM MEM

PROM MEM has 4 control lines. They are hardwired as follows:



## Hardwiring PROM WB

PROM WB has 3 control lines. They can be hardwired as follows:



## CONCLUSION

Hands-on education is usually the most effective. The ability to build processors using hardware gives students an invaluable experience. Although some may find it more time consuming in comparison to simulation, our experience shows that the hours the students spent building their own processor really pays off. Students pay attention to the minute details and hence end up with a solid understanding of processor design. Our goal is to provide an instruction set architecture that is simple enough so students can build it physically using hardware. Yet realistic enough so it gives insight on how real processors work. In this paper we continue our previous work and provide details of the most complex part of any processor design: the control unit. We showed two implementations for the control unit; one is microcoded and the other is hardwired.

## References

1. Linda Null, *The Essentials of Computer Organization And Architecture*, Jones and Bartlett publishing, 2006.

2. Patterson and Hennessy, *Computer Organization and Design: The Hardware/Software Interface*, Morgan Kauffman, 2007.

3. M. Holland, J. Harris, and S. Hauck, "Harnessing FPGAs for Computer Architecture Education", *Int. Conf. on Microelectronic Systems Education*, June 2003.

4. Nestor J., "Teaching Computer Organization with HDLs: An Incremental Approach", Proceedings of the 2005 IEEE Int. Conf. on Microelectronic Systems Education.

5. Calazans and Moaraes, "Integrating the Teaching of Computer Organization and Architecture with Digital Hardawre Design Early in Undergraduate Courses", IEEE Transactions on Education, v.44 no. 2, pp.109-119, 2001.

6. Ellard et al. "On the design of a new CPU architecture for pedagogical purposes", Proceedings of the 2002 workshop on Computer architecture education: Held in conjunction with the 29th International Symposium on Computer Architecture, May 2002.

7. Vollmar K., "MARS: an education-oriented assembly language simulator", Proceedings of the 37th SIGCSE technical symposium on Computer science education, March 2006.

8. Computer System Architecture Lab, http://yuntenlabs.com/csalab.htm.

9. ElAarag H., "A Complete Design of a RISC Processor for Pedagogical Purposes", to appear Journal of Computing Sciences in Colleges.

## Biographical Information

Hala ElAarag received her Ph.D. degree in Computer Science from the University of Central Florida, Orlando, in 2001. She is currently an associate professor of computer science at Stetson University, DeLand, FL. Her research interests include computer architecture, network performance, Internet protocols, wireless networks, network simulation, and operating systems. Dr. ElAarag had obtained Stetson University research award in 2005 and Best Paper Award at the 11th Communication and Networking Symposium (CNS'08). She was co-general chair of Communication and Networking Simulation Symposium 2009. She serves on the technical committee for many international conferences and reviews for multiple journals.

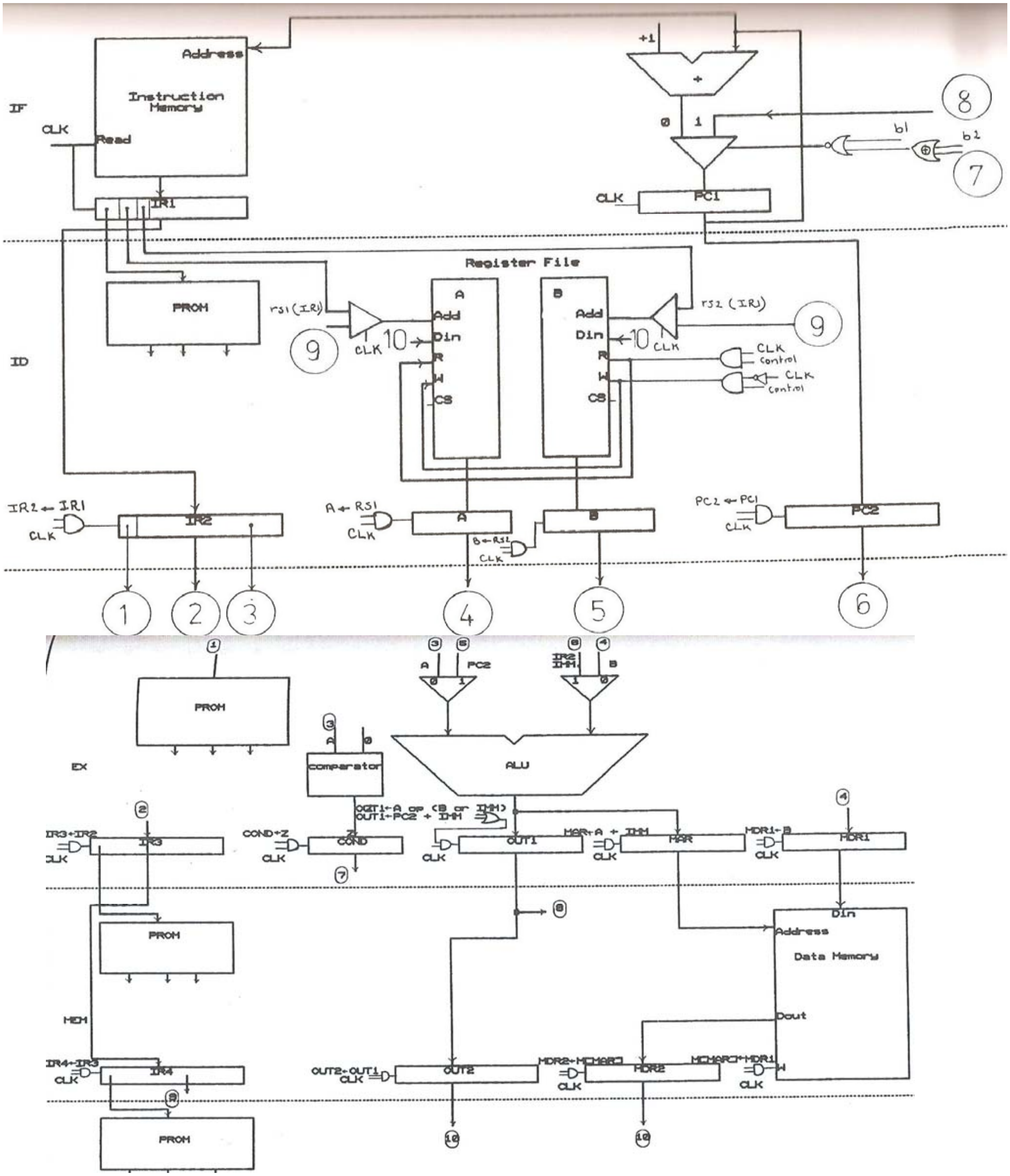


Figure 2: Pipelined HRISC Block Diagram.