Volume 4, No. 4, March-April 2013

ISSN No. 0976-5697

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

J1 Accumulator-Based Processor for Educational Purposes

Askari Yadollahpour Mansour
Department of Computer
Science and Research Branch, Islamic Azad University
Tabriz, Iran
as.yadollahpour@yahoo.com

Ahmad Habibizad Navin
Department of Computer
Science and Research branch, Islamic Azad University
Tabriz, Iran
habibi@iauotash.ac.ir

Amir Masoud Rahmani
Department of Computer
Science and Research Branch, Islamic Azad University Tehran, Iran
rahmani@srbiau.ac.ir

Abstract: Computer architecture courses are usually necessary in the first years of computer science and computer engineering degree, as learning the computer architecture has influence on learning other computer concepts such as Operating system, compiler and so on. A simple computer with complete instruction set can be used to enable the students to get required skills, so a simple 8-bit computer is designed to teach basic concepts and principles of the computer. Having designed the stage, to show the correct functionality of the proposed processor, we modeled the processor with Verilog HDL and simulated it with SynaptiCAD Verilogger Extreme software. We realized that by using the proposed processor in computer architecture education, basic concepts are taught simply and quickly.

Keywords: computer architecture; CPU; educational; processors; accumulator-based; one address machine

I. INTRODUCTION

Computer architecture and organization is one of the key knowledge areas in Computer Engineering [1]. Learning the computer architecture has significant influence on the learning of other courses related to the computer such as Operating system, compiler and so on. The computer architecture course should achieve multiple objectives. It must provide an overview of computer architecture and teach students the operation of a typical computer. Also it should reinforce topics that are common to other areas of computer science; for example, teaching register indirect addressing reinforces the concept of pointers in C language [1].

Our focus in this paper is on how to teach the subjects related to the computer fundamentals courses cover instruction format, instruction set architecture, data path design, memory and addressing modes. A simple educational processor can be used in these courses to educate basic computer concepts. Some of the educational machines derived from real computers such as miniMIPS [2], Sap 2, 3 [3] and Ant 32 [4], [5], are not suitable to be used in education because of their high complexity. On the other hand some processors such as Ant-8 [6], [7] and YASP [8] have considerable education-specific properties but are relatively complex. SAP 1 [3] has very simple instruction set, as it doesn't support loop instructions. Nonetheless, teaching its hardware details is somewhat difficult because its control circuit needs six T sates to execute one instruction that makes control circuit design difficult.

The main idea of this paper is to design an educational processor for teaching computer architecture courses for the following purposes:

a. Covering the basic concepts,
b. Teaching simpler and faster.

To acquire the above purposes the new educational processor must use new educational tools and technology. So the main purpose of this paper is to design a simple computer with relatively complete instruction set and to develop a visual simulator for teaching the basic concepts faster.

The rest of this paper is organized as follows: in section II, educational processors are reviewed, in section III, new educational processor is introduced, in section IV simulation results is discussed. Section V concludes the paper.

II. REVIEW OF EDUCATIONAL PROCESSORS

A. Basic Introduction:

Some important educational processors are reviewed in this section. Ant-8 and Ant-32 are two RISC-Based educational processors presented at Harvard University [4] , [5] , [6] , [7]. Ant-8 is 8-bit and suitable for computer fundamental courses but Ant-32 is too complicated to be used in elementary computer architecture courses. Sap-1, Sap-2 and Sap-3 are three Accumulator-based processors that have been introduced by P. A. Malvino in 1977. Sap-1 is a very simple and suitable for teaching preliminary concepts. Sap-2 and Sap-3 are real-based processors, and suitable for teaching operating system, assembly programming, microprocessor and compiler courses. M. mano presents a 16-bit accumulator-based processor in 1976 that named Mano's basic computer. This processor has complete instruction set and supports two hardware interrupts. Mano and Kime also introduced two RISC educational processors in 1997 which respectively are named:

Mono's simple computer (MSC) and Mano's pipelined RISC CPU (MPRC) [10]. MPRC is an advanced version of MSC with a more complete instruction set and pipeline implementation. MPRC is one of the best cases for studying pipeline concepts.

ASC is an accumulator-based processor that introduced by Shiva in 1985 [11]. Shiva presents Full software and hardware design details for ASC. YASP educational processor was introduced since in the academic year 1999-2000, step by step [8]. The same as ASC, YASP supports various addressing modes and similar to micro controllers has various I/O ports. VeSPA is a 32-bit processor proposed in the book "Designing Digital Computer Systems" in 2005 [12]. Its instruction set is similar to MIPS32 and supports more addressing modes. Pipelined version of VeSPA is also proposed in [12]. Sweet-16 is a 16-bit fully functional single-cycle RISC processor developed in Heidelberg University within 2000-2008 for the illustration and use in computer architecture courses [13]. It has complete instruction set, and has optimized architecture for pipeline implementation.

B. Summary:

Some important educational processors were briefly reviewed in this section. Some of the educational machines derived from real computers such as Sap 2, 3 and Ant 32 are not suitable to be used in education because of their high complexity. Accumulator-based educational processors such as YASP and ASC are one address machines that follow the von Neumann architecture. They have an accumulator and one or more auxiliary registers in the data path to perform arithmetic or logical operations. Some of them such as YASP have complex instruction set. They are simple and good for education in elementary courses. RISC-based educational processors such as sweet-16 and ant-8 follow Harvard Architecture. Each of them has a register file in the data path and most of processing operations is done by register-to-register instructions. These processors are suitable for teaching, but they are not quite suitable for beginners.

III. J1 PROCESSOR

A new educational processor, J1, which is easy and simple for beginners, is designed in this section. J1 is accumulator-based processor but such as RISC processors only uses load and store instructions to access the main memory. This processor compared to SAP1, has more complete instruction set and covers more concepts. Yet, its control unit needs only three timing states to operate. This makes the control circuit design very simple, also simpler to educate. J1 is designed to achieve the following purposes:

a. It is Easy to teach and understand,
b. It has a relatively complete instruction set

Table I shows the instruction set of the j1 processor. J1 includes 11 instructions but has room to extend its instruction set to 16. Figure 1 shows instruction format for the processor. It has only one format type and supports only direct addressing mode.

A. J1 visual simulator:

For a profound and quick training, a visual simulator has been designed for J1. This simulator helps to simulate program execution and assembly language programming and debugging program. Figure 2 shows visual simulator for the J1 processor. The simulator user can write his program, then assemble it into memory, trace execution and see the results in the registers. Through this approach, teaching fundamental concepts is done more quickly.

B. Architecture:

Figure 3 shows the architecture of J1. The processor has a bus-based multi-cycle data path. The registers PC, IR and AR are 4 bit and others are 8 bit. Bus is 8 bit. Each instruction requires three timing signals; T0 to T2 to run. For example the

Table 1. Instruction set for the J1 processor

Opcode	Mnemonic	Operation
0000	ADD A, B	A ←A + B
0001	SUB A,B	A ← A - B
0010	AND A,B	A ← A & B
0011	OR A,B	A ← A | B
0100	IN address	A ← IO[address]
0101	OUT address	IO[address] ←A
0110	LDA address	A ← M[address]
0111	STA address	M[address] ← A
1000	JMP address	PC ← address
1001	JNZ address	If Z!=0 then PC ← address
1010	LDB address	B←M[address]
1111	HLT	Stop Machine

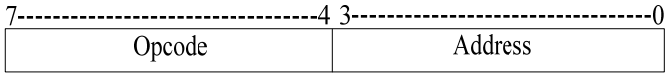

Figure 1. Instruction format of J1 processor

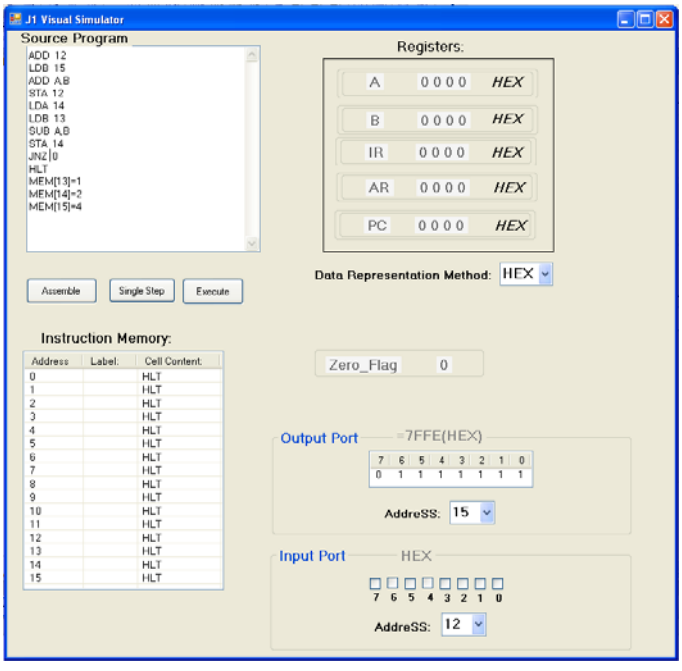

Figure 2. J1 visual simulator

following RTL shows how the ADD instruction is fetched and executed during T0 to T2 states:

T0: AR←PC

T1: [IR , AR] ←M[AR] ;PC++

T2: A←A+B

All instructions operate the same during T0 and T1 states. The stage that occurs during T0 to T1 states is known as the fetch stage. The decode stage starts immediately after loading new opcode to the IR (Instruction Register). In this stage a 4*16 decoder decodes opcode of instruction that being executed. The execution stage occurs during T3 state. Table II shows control unit operation and active signals to execute each instruction during states. This machine has two ports: port number 12 and port number 15). The port 12 is used for input data from keyboard and the port 15 is used to display the output data.

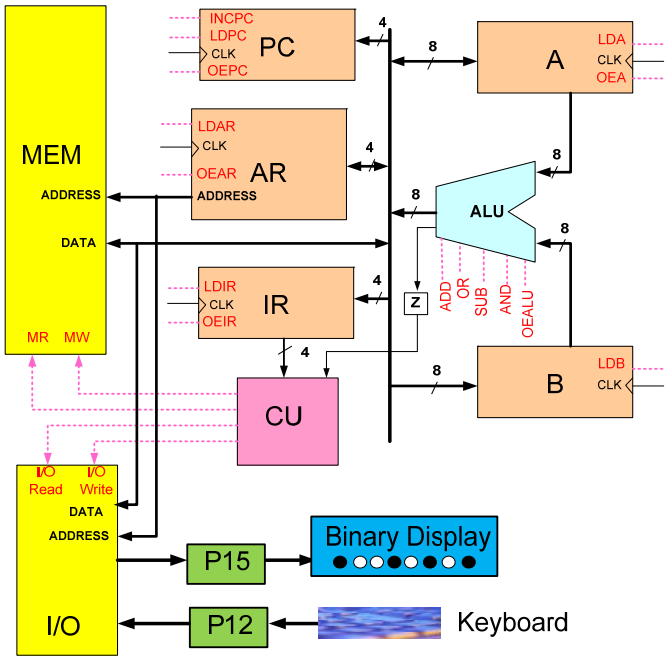

Figure 3. J1 architecture

Table 2. Control unit operation in each state

State	*Register transfer*	*Active signals*
T0	AR←PC	LDAR,OEPC
T1	[IR , AR] ←M[AR] ;PC++	LDIR,LDAR,INCPC,MR
T2.ADD	A←A+B	ADD,OEALU,LDA
T2.SUB	A←A+B	ADD,SUB,LDA
T2.AND	A & B	OEALU, AND,LDA
T2.OR	A | B	OEALU,OR,LDA
T2.IN	A←I/O	LDA,IOR
T2.OUT	A←M	OEA, LDA
T2.LDA	A←M	MR,LDA
T2.STA	M←A	OEA,MW
T2.JMP	PC←address	LDPC,OEAR
T2.Z'.JNZ	PC← address	LDPC,OEAR
T2.HLT	Stop	STP Counter

C. Control unit for J1:

Control unit of J1 was designed by using hard-wired approach. Block diagram of J1 control unit is shown in figure 4. Input to the controller consists of the 4-bit opcode of the instruction currently contained in the instruction register and the zero flag from the accumulator. The controller's output goes out to the various registers and to the memory of the computer.

The control unit is composed of 3 units: an instruction decoder, a state generator and a control matrix. State generator provides a sequence of three consecutive active signals that cycle continuously. Synchronized by the system clock, the state generator first activates its T0 line, then its T1 line, and finally its T2 line. After T2 is active, the sequence begins again with T0. The instruction decoder takes its four-bit input from the opcode field of the instruction register and activates one and only one of its 19 output lines. Each line corresponds to one of the instructions in the computer's instruction set.

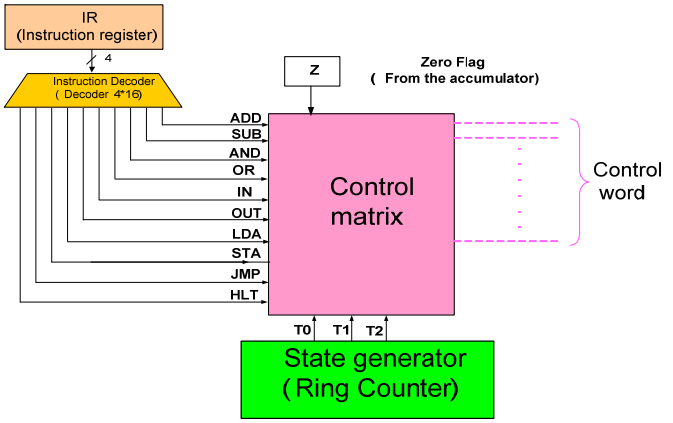

Figure 4. Hard-wired control unit for J1

IV. SIMULATION OF J1

J1 has been modeled by verilog codes simulated in SynaptiCAD Verilogger Extreme software. Figure 5 shows the simulation results of the following code that calculates 4*2, along with suppose that the numbers 4 and 2 has been stored respectively at address 15 and 14 of the memory and the number 1 stored at address 13 of the memory:

```
LDA   12
LDB   15
ADD   A,B
STA   12
LDA   14
LDB   13
SUB   A,B
STA   14
JNZ   0
HLT
MEM[13]=1;
MEM[14]=2;
MEM[15]=4;
```

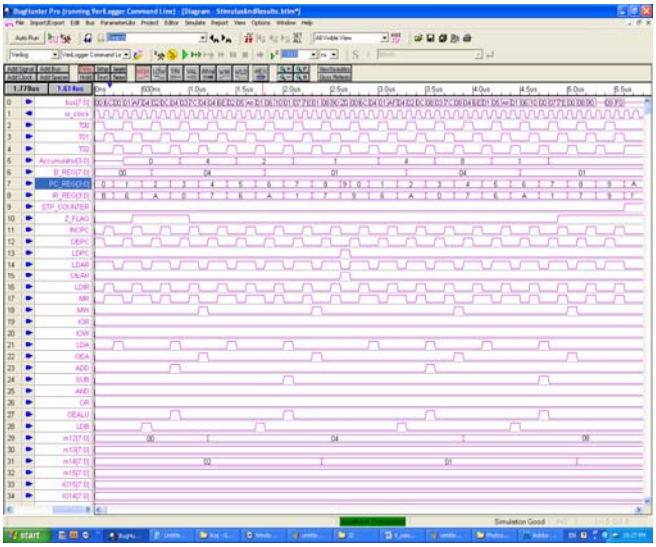

Figure 5. Simulation of a sample program to calculate 3*5

V. CONCLUSION

In this paper we proposed a simple computer, J1, that can be used to enhance computer architecture education. J1 is an accumulator-based processor and has very simple architecture. J1 architecture is simulated with verilog HDL that enables students to practice designing the components of computer systems and explore architectural concepts. Also a visual simulator was designed for J1 that contains an integrated assembler, debugger and simulator. Simulator user can write his program, then assemble it into memory, trace execution and see the results in the registers. By this approach students can learn computer architecture more quickly and profoundly.

VI. REFERENCES

[1] B. Nikolic, Z. Radivojevic, J. Djordjevic, and V. Milutinovic, "A survey and evaluation of simulators suitable for teaching courses in computer architecture and organization", ;presented at IEEE Trans. Education, 2009, pp. 449-458.

[2] D. A. Patterson and J. L. Hennessy, Computer Organization and Design: The Hardware/Software Interface, 4th ed., Morgan Kaufmann, 2008.

[3] A. P. Malvino, Digital Computer Electronics, Gregg Division, McGraw-Hill, 1977.

[4] D. Ellard, D. Holland, N. Murphy, and M. Seltzer, "On the design of a new CPU architecture for pedagogical purposes," in Proc. Workshop Comput. Architecture Edu., Anchorage, AK, May 2002, pp. 28–34.

[5] D. Ellard and P. Ellard, ANT-32 Assembly Language Tutorial (for version 3.3.1b) 2003 [online]. Available: http://ellard.org/dan/www/pubs.

[6] D. Ellard, P. Ellard, J. Megquire, J. B. Chen, and M.Seltzer, The ANT Architecture — An Architecture for CS1, The IEEE Computer Society Technical Committee on Computer Architecture Newsletter, February, 1999, 25–27.

[7] ANT-8 3.3.1b Assembly Language Tutorial 2003 [Online]. Available: http://ellard.org/dan/www/pubs.

[8] L. Ribas-Xirgo, "Yet another simple processor (YASP) for introductory courses on computer architecture," IEEE Trans. on Industrial Electronics, vol. 57, no. 10, pp., Oct 2010.

[9] M. M. Mano, Computer System Architecture, 3rd ed., Prentice Hall, 1992.

[10] M. M. Mano and C. R. Kime, *Logic and Computer Design Fundamentals*, 3rd ed., Prentice Hall, 2003.

[11] S. G. Shiva, Computer Organization, Design and Architecture, 4th ed., CRC Press, 2008.

[12] David J. Lilja and Sachin S. Sapatnekar, Designing Digital Computer Systems with Verilog, Cambridge University Press, 2005.

[13] V. Angelov and V. Lindenstruth, "The educational processor Sweet-16," in Proceedings of the 19th International Conference on Field Programmable Logic and Applications (FPL '09), pp. 555–559, August 2009.