Machine Cycle CPU Simulator for Educational Use based on Squeak Environment

Takao KAWAMURA, Yoshio KAWAGUCHI, Shinji NAKANISHI, and Kazunori SUGAHARA
Department of Information and Knowledge Engineering, Faculty of Engineering, Tottori University
{kawamura,sugahara}@ike.tottori-u.ac.jp

Gen SUZUKI
Tottori Prefectural Industrial Reserch Institute

Abstract

A machine cycle CPU simulator is developed on the Squeak environment for educational use. The developed simulator is able to show hardware behavior in CPU at each system clock. Any component of the simulator is implemented as an Morphic object in Squeak. The developed simulator is examined by execution of example programs and correct behaviors of their executions are confirmed.

1. Introduction

Varieties of high-level programming languages and their development environments have been proposed based on the highly integrated and high performance CPUs. Usually, in such high-level programming environments, hardware behaviors of CPU circuits are hidden behind as abstract. However, performance of software and that of hardware are inextricably linked. In educational establishments such as high schools or universities, it is desirable to teach software and hardware of computer technologies comparably for educating software engineers. But it becomes difficult to teach how instructions and data are executed and flows in CPU circuits. Considering these points, a machine cycle simulator of virtual 16 bits CPU is developed on the Squeak environments [1] to show the data flow in execution of programs. We have selected Squeak as the development environment for our simulator because of its interactiveness and its user-interface framework, Morphic.

2. Target CPU – SIMPLE –

SIMPLE(SIxteen bit MicroProcessor for Labolatry Experiments) is a virtual CPU designed by Tomita and Nakajima for experiments and educational use in laboratories or classrooms[2]. Its characteristics are summarized as follows,

1. 16 bits CPU,
2. $2^{16}(= 64k)$word memory area,
3. Base register addressing method,
4. Every machine code has one word length,
5. Following 7 categorized 28 instructions,
   (a) Load instruction,
   (b) Store instruction,
   (c) Immediate load instruction,
   (d) Unconditional branch category including 3 instructions,
   (e) Conditional branch category with 8 instructions,
   (f) Arithmetical and logical operation category with 11 instructions,
   (g) Control category with 3 instructions.
6. Every instruction is executed in 5 phases of,
   (a) Phase 1: Instruction fetch phase,
   (b) Phase 2: Register loading phase,
   (c) Phase 3: Operation phase,
   (d) Phase 4: Main memory access phase,
   (e) Phase 5: Register storing phase.

3. Functions of Simulator

The proposed simulator has following functions to perform clock level machine code simulation.

1. Show activated circuit blocks in every execution phase,

2. Show data-flow among registers in CPU and main memories in every execution phase,

3. Flexible user interface, such as program loading function from files.

4. Software implementation

A screen image of the proposed simulator is shown in Fig.1.

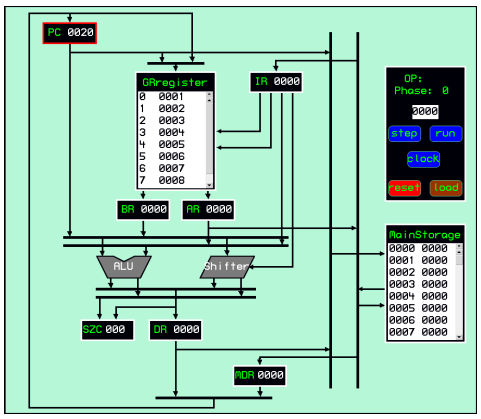

Figure 1. Screen image of the proposed simulator.

The developed simulator is composed of a base, a control-unit, registers, register-collections, data-lines and buses. Each component is implemented as an object in Squeak and has its own view on the screen.

The base of which view is shown as gray background board in Fig.1, creates and manages all other components. The control-unit sends phase signals to activate components and its view shows the current phase status and executing opcode. It also has a controller with which a user generates system clock signal and reset signal through mouse clicks.

The rest of the components, having one or more preceding components and one or more succeeding components, receive data from the preceding components, process them if necessary and send them to the succeeding components.

These components act upon two type of messages, i.e. phase signal messages from the control-unit and data messages from the preceding components. When a component receives the first type of message, it determines whether data in its data buffer must be processed immediately according to the current phase status and the opcode obtained from the instruction register (IR). If data must be processed immediately, the components process them and send the result to the succeeding components and highlights own view.

When a component receives the second type of message, it determines whether data included the message must be processed immediately by the same manner as the case of the first type message is received. If data must be processed immediately, the components process them and send the result to the succeeding components and highlights own view. Otherwise, the component saves them into its data buffer.

5. Experiments

The developed simulator is examined by execution of example programs. In the first example, an integer stored in the main memory is simply loaded into certain register. The mnemonic code of this example is shown in Fig.2, and correct behavior of its execution is confirmed.

`LI`	`1,10`	$r[1] = sext(10)$
`LD`	`0,8(1)`	$r[0] = *(r[1] + sext(8))$
`HALT`		$halt()$

Figure 2. Example program 1.

The second example is calculation program. Summation of integer from 1 to 5 is achieved and the final result is stored in register 3. The mnemonic code is shown in Fig.3 and it is confirmed that this simulator is able to get correct answer.

`LI`	`0,5`	$r[0] = sext(5)$
`LI`	`1,1`	$r[1] = sext(1)$
`MOV`	`2,0`	$r[2] = r[0]$
`SUB`	`0,1`	$r[0] = r[0] - r[1]$
`BLE`	`2`	$if(Z\|\|S) PC = PC + 1 + sext(2)$
`ADD`	`2,0`	$r[2] = r[2] + r[0]$
`B`	`-4`	$PC = PC + 1 + sext(-4)$
`HALT`		$halt()$

Figure 3. Example Program 2.

6. Conclusion

In this paper, the machine cycle CPU simulator is developed on the Squeak environment for educational use. The developed simulator is able to show hardware behavior in CPU at each system clock.

References

[1] G. Korienek, T. Wrensch, and D. Dechow. *SQUEAK – A Quick Trip to Object Land –*. Addison-Wesley, 2002.

[2] S. Tomita and H. Nakajima. *Computer hardware*. Shokodo CO., LTD., 1996. in Japanese.