MKit Simulator for Introduction of Computer Architecture

Seikoh Nishita
Department of Computer Science,
Faculty of Engineering, Takushoku University
snishita@cs.takushoku-u.ac.jp

Abstract

For the introduction of computer architecture in computer science, highly simplified specification of CPU, and visualization of CPU operation are important. This paper describes a CPU simulator we have developed. It supports students' learning of introduction of computer architecture. It presents internal status of CPU by processor block diagrams, and provides buttons for manipulation in various ways. And our simulator provides set of the diagram and the buttons as the progress of students' learning.

1. Introduction

The introductory education on the computer architecture involves several topics: about the relationship between assembly and machine language, the connection of data-path and data-path elements as the static construction of a processor, the operation for instructions as dynamic structure, and the control by control unit and control signals. These topics focus different aspects of the processor. For example, instruction set is focused on the topic about the relationship of languages. On the other hand, processor block is focused on the topics of static and dynamic structure of CPU. And Control unit and control signals are focused on the topic about control.

This paper describes a CPU simulator, "MKit simulator" that we have developed. MKit simulator has several interfaces for students to check and manipulate MKit processor. These interfaces are draw panels in the simulator window, and input methods like buttons. The draw panel shows inner state of the processor, and the input methods are used to manipulate the processor and the simulator. These interfaces emphasize respectively various aspect of the processor. We correspond these interfaces with the topics that have same aspect of the processor. Our simulator provides students a part of the interfaces for every assignment for the corresponding topic. Since there are a number of topics students learn during an exercise, our simulator switches interfaces to use for every assignment.

The next section gives the specification of CPU for our simulator. Section 3 then describes simulator designing. Section 4 describes assignments we have designed for an exercise class with our simulator. Section 5 then describes implementation of the simulator. Section 6 discusses the relation of our simulator and other simulators that have been proposed for introductory education of computer architecture.

2. Specification of MKit

Before we discuss the designing of MKit processor, we describe our background of computer architecture education in our department. We have one course on computer science, and there are three classes closely concerned with computer architecture. The first is a class on introduction of computer science, where first year students learn basic concept of computer architecture with a simple CPU, MKit [4]. The second is a basic exercise on computer science, where our simulator is used. And the third is a class on computer architecture, where students learn advanced. The goal of the basic exercise class is students' making certain of the basic concept of computer architecture that they have learnt at the first year class.

We adopt MKit as the processor for our simulator. In deciding the processor we take care to keep the design as simple as possible, and to link the topic of first year class and the exercise class closely. When students are going to have the basic exercise class, they have learnt computer architecture with MKit processor. Therefore students are familiar with MKit, and they don't take relatively long time learn the specification of MKit than new specifications. As the result they can concentrate their thought to essential of computer architecture and assignments in the exercise.

MKit CPU is a 16-bit word accumulator machine. It has only the direct addressing as the addressing mode to access memory. The memory access is 16-bit word addressable. Making the machine word addressable only simplifies the operation of CPU. Moreover care was taken to keep the correspondence between assembly language instructions and the machine instruction as one-to-one relationship. As the result, the address width is reduced to 12 bits, and the operation code occupies 4 bits of a word. This configuration simplifies the hexadecimal number representation in the machine language. MKit only supports two instruction formats as shown in Figure 1.

The instruction set of MKit consists of arithmetic calculations, jump/branch operations and memory access (load and store) operations (as shown in Table 2). It has instructions for a subroutine call “JL” and a return operation “RET”. These instructions make the processor complicated, but they are straightforwardly integrated into the instruction set and the operation of the processor. Moreover they encourage students' learning of the concept on subroutine call operation without stacks.

The data-paths of the processor are not based on bus structure. The data-paths connect the data-path elements directly. And the instructions take multiple clock cycles to execute. Data are roughly flown clockwise along with data-path in a processor block diagram shown in Figure 3. One difference from other processors for introduction of computer architecture is that the processor has a return address register, RAR. RAR is used for the subroutine call and return operation. Since RAR is not stack, this micro-architecture does not support multiple subroutine call. But the micro-architecture shows an implementation for subroutine call, and it can be simply extended by replacing a register with a stack for RAR.

3. Simulator Designing

The basic concept of computer architecture for introductory education consists of several topics. On this paper, we divide the basic concept into following 3 topics.

- Instruction set architecture, and translation of assembly language instructions to machine language instructions
- Data-path connection with data-path elements (static construction of the processor except control unit), and the operation of the processor: instruction fetch and execution (dynamic construction)
- Control unit and control signals. Static and dynamic construction of the processor.

We also assume that assignments are given along with these topics.

The processor is placed as various objects on these topics respectively. For example, on the topic about instruction set architecture, the processor is regarded as a machine that executes instructions. For the topic about data-path connection, the processor is regarded as a structure represented by the processor block diagram. On the topic about control unit, the processor is regarded as a structure represented by the diagram with the control unit and lines for control signals.

Moreover these topics have their adequate assignments, which need functions to manipulate the processor. For example, functions for execution all instruction and execution one instruction are helpful to make certain of the meanings of instructions. Functions for step-by-step execution is useful to learn the operation of the processor.

Interfaces of CPU simulator emphasize the aspects of the processor, and they give the functions

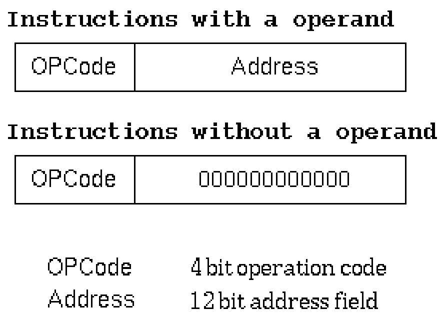

Figure 1. Instruction encoding formats

Sort	*Nmenonic*	*Semantics*
Memory access	LD n	ACC <- Mem[n]
	ST n	Mem[n] <- ACC
Arithmeti c n	ADD n	ACC <- ACC+Mem[n]
	SUB n	Mem[n]
	SFR	ACC <- ACC>>1
	SFL	ACC <- ACC<<1
Jump and Branch	JMP n	PC <- n
	JPZ n	PC <- n (if ACC=0)
	JPN n	PC <- n (if ACC<0)
	JPO n	PC <- n (overflow)
	JPL n	PC value is kept, then PC<-n
	RET	restored
Control	HLT	halt

Table 2. Instruction Set

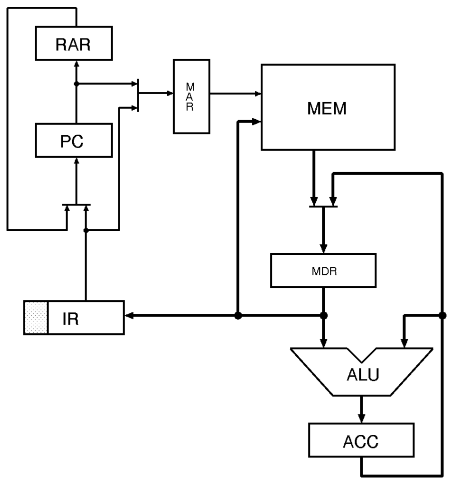

Figure 3. Processor Block Diagram

to manipulate. The interfaces are divided in interfaces to express inner status of the processor, and interfaces to manipulate the processor. These interfaces are often implemented as drawing panels or push buttons in a same window.

Our idea for designing a simulator is based on multiple uses of the expression interfaces and manipulation interface, and it is based on a concept of correspondence between the interfaces and the topics. Since the processor is regarded various object as the topics, we adopt the expression interfaces in order to represent the aspects of the processor. We also select the manipulation interfaces in order to design assignments.

On this paper, we design following manipulation interfaces from M1 to M5.

M1 is for execution all instructions in memory unit. It is implemented with one button, which activates virtual processor to run a program one by one along with clocks. This process stops till the processor decodes the halt instruction. This interface is suitable, when the processor is regarded as a machine to execute instructions.

M2 is for execution one instruction in memory unit. It is same as M1 except the fact that it activates virtual processor to run only single instruction.

M3 is for execution one-step of the process. It is same as M1 except the fact that it activates virtual processor to run only one step (one clock). Since one step of the execution may not be represented in semantics of the instructions, this interface is not suitable for students to check instruction architecture. But it is suitable for checking the inner operation of the processor, because it helps students to analyze the operation in detail.

M4 is for assertion of control signals directly. It is implemented with buttons, which assert all control signals in the processor. This interface is suitable for the topic about control unit and signals. It helps students learning of control signals and deriving their sequence of the fetch and execution operation.

M5 is for making time chart of control signals for instruction fetch and execution operation of all instructions, in order to specify the control unit. The branch instructions have two time charts for the condition of the branch. This interface is also suitable to the topic about the control unit and signals. Since we assume that students don’t learn the subject of the logic circuit designing, there is no interface for designing control unit with logic circuit. Instead of logic circuit, we adopt time charts. Though we need to show how to interpret and write time charts, it doesn't take longer time for explanation about time chart than about logic circuit designing. Moreover we take advantage of supporting education by specifying the control unit. That is, combination of the interface M5 and M2 facilitates students to learn that the control unit generates the sequence of control signals automatically and these signals activate the operation of the processor.

We also use auxiliary interfaces to reset the processor, to suspend the execution and to adjust the speed of the execution.

In order to present the inner state of the processor, we make visible/invisible attributes for sorts in the processor block, i.e. control unit, path of control signals, data-path and data-path elements. By these attributes, objects are shown/hidden on the processor block diagram. Following list shows the attributes of the every sort in the processor.

- Data-path: “data-path” and “data-flow” attributes. The former is for static representation of data-paths, and the latter is for dynamic representation of flowing data.
- Data-path elements: “elements”, “value”, and “flag” attributes. The “value” attribute is used to show data kept at registers.
- Control unit and control signal: “control” attributes. The “control” attributes is used to show the both of the control unit and the lines of control signals.

We design the expression interfaces from E1 to E5 by specifying all attributes of the sorts in the processor block. The interfaces E1, E2 and E3 are shown at Figure 4, 5 and 6 respectively.

E1: “data-path-flow”, “elements”, “value” and “flag” of primary data-path elements are visible.
E2: all attributes except "control" are visible.
E3: all attributes are visible.
E4: all attributes except "control" are visible.
E5: all attributes except "control" and "value"

As addition of these interfaces, we also use a message box for explanation of the operation. In order to design the expression interface we define the word "primary data-path elements" for elements referred in the semantics of the instruction set architecture. The primary data-path elements of MKit are the program counter, the instruction register, the accumulator, and the memory unit.

All attributes specify simply their visibility, that is, if an attribute of a sort is visible, objects of the sort are always drawn in same position of the simulator window. This specification of the expression interface makes it simple to represent the inner status of the processor and makes it easy for students to grasp the operation.

4. Interface and Assignment Design

We introduced interfaces to express status and manipulate CPU simulator in previous section. In practically use of the simulator, the expression interfaces and the manipulation interfaces are combined along with assignments for the topics of computer architecture. We discuss with the assignments and combinations of interfaces we have designed. The assignments are assumed to given in the following order.

4.1 An assignment about the instruction set architecture

On this assignment, students translate assembly language instructions to machine language instructions. Then students make certain of the semantics of the instructions they have learnt. During the assignment, students write instructions into memory unit of our simulator, and execute instructions by MKit virtually.

We choose the interface E1, M1 and M2 for the assignment. Since the interface E1 draws only flowing data and primary data-path elements, E1 facilitate students to concentrate the semantics of instructions during the assignment. And because E1 does not draw whole processor block diagram, it is also suitable for students who have not learnt the processor block as micro architecture yet.

The interface M1 and M2 is used to execute instructions. These interfaces make students to confirm the semantics and a part of the processor operation, by checking flowing data on the simulator window. We note that the there is not the interface M3, which is used for executing step by step along with the clock. Since there are steps that don’t bring any change of the primary data-path elements, the step-by-step execution is not suitable for the interface E1.

4.2 An assignment about the processor block

On this assignment, students confirm the static structure of the processor by the processor block diagram. Then students execute instructions in the memory unit, and make certain of the processor operation on the processor block diagram. This assignment supports students to confirm that there are registers and data-path among the primary data-path elements in order to propagate data.

We choose the interface E2, M1, M2 and M3 for the assignment. The interface E2 draws the processor block diagram except the control unit. E2 makes students concentrate to learn the processor block. Students can also execute instructions and check the operation of the processor by the interface M1. When students want to check the detail of the process, they can use M2 and M3 too.

4.3 An assignment about the control unit

We design three assignments for the topic on the control unit: an assignment about the static construction of control unit, two assignments about the operation of the control. These three assignments are described from this section to the section 4.5.

On this assignment, students make certain of the control unit and the lines of control signals. The goal

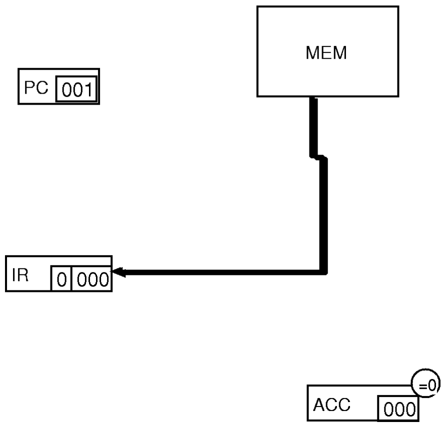

Figure 4. Interface E1

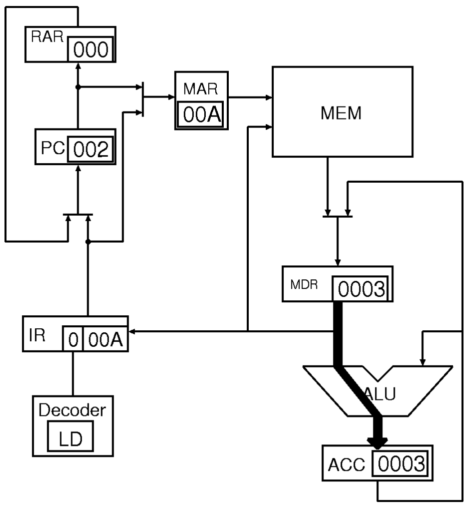

Figure 5. Interface E2

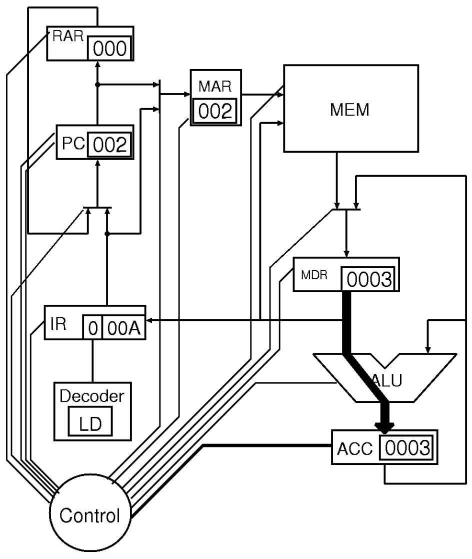

Figure 6. Interface E3

of the assignment is that students confirm the existence and roles of the control unit and control signals.

We choose the interface E3 for the assignment. The interface E3 shows all sorts of the processor including the control unit and the lines of control signals. The minutest diagram of processor block makes students to learn and remember that the process of CPU arises from the control unit and signals.

Since there is no interface for manipulation, all students can do is to confirm the static construction of the processor block diagram. The next assignment makes demands of students for more active learning.

4.4 An assignment to control CPU by asserting control signals directly

On this assignment, students try to derive the sequence of control signals for fetch and execution of all instructions. This assignment helps students make certain of the instruction fetch, decode, and execution in the CPU process, through this assignment.

We adopt the interface E4 (or E5) and M4. In order to assign student to derive the sequence of control signals, we make students assert the signals via buttons provided by M4. Figure 7 shows the interface E5 with M4. The buttons of control signals (“load”, “+1”, “write” and so on) enables students to assert signals. In order to execute instructions, they derive the correct sequence of control signals on the basis of their knowledge they have learnt the first 2 assignments.

The interface E4 draws the processor block diagram except the control unit and the lines for signals. We indicate the lack of the control unit, and students have to perform the role of the control unit.

The use of the interface E5 makes this assignment more difficult, because E5 does not show values kept by registers. Therefore students check only OPcode and flags, and then they assert control signals with the use of E5. The interface E5 helps students to learn the control unit has only these values as their input.

The interface M4 provides buttons for all control signals. The buttons are drawn of the processor block diagram at the draw panel. Drawing the buttons on the diagram makes manipulation of the buttons intuitive.

4.5 An assignment to control CPU with time chart

On this assignment, students make time charts for fetch and execution of all instructions. Then they execute instructions in the memory unit with the time charts they specified. The goal of this assignment is that students summarize the sequences of control signals, and they confirm the role of control unit that works automatically.

The virtual processor in the previous assignment was incomplete in the sense of the control unit absence. On this assignment, students make the processor complete by the time charts for the control unit.

We choose the interface E3 and M5. The interface E3 is used in order for students to check the values of control signals, and to make it easy to try and error for specifying time charts. The interface M5 shows time charts of all control signals for fetch and execution of all instructions. And it accepts mouse clicks to modify signals(Figure 10).

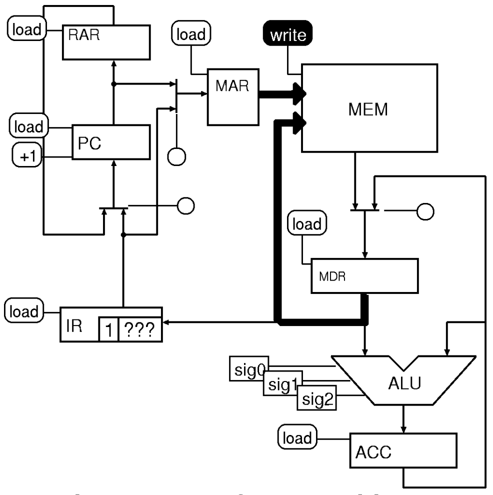

Figure 7. Interface E5 with M4

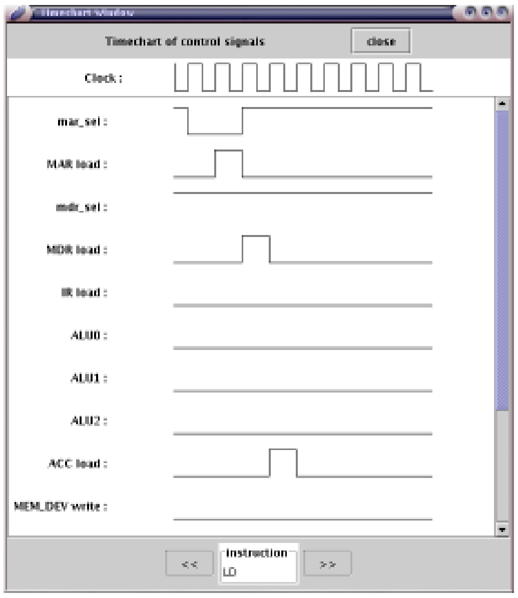

Figure 10. The Interface to Make Time Chart

5. Simulator Implementation

We have implemented MKit simulator with Java 1.4. Our implementation is based on object-oriented concept, where the parts of the processor (data-path, data-path elements, control units, control signals) are treated as objects. Each object has its attribute for the expression interface. And each object has methods to implement the manipulation interfaces.

Objects have their own positions on the simulator window respectively. If the current interface is changed, objects are drawn/hidden at the positions according to the attributes. As an advantage of this behavior of object, it makes students' grasp of CPU status easier, because a part of the single processor block diagram is always drawn, and no object changes it's position among the interfaces.

In order to design other exercises and assignments, we can change the order of the interfaces and the attributes of the interfaces. But in order to change the interfaces, we need to modify Java programs on current status of our implementation. There is other root for refinement of the current implementation. That is, our source package will be divided into three modules, i.e. the virtual processor, interfaces for expression and interfaces for manipulation. This refinement will make the simulator more flexible and highly extendable for modification of the interfaces and the virtual processor.

6. Related Works

In order to support computer architecture education, several CPU simulators and simple specifications of CPU have been proposed [1,2,3]. These simple specifications are designed deliberately for first or second-year students to learn essential of computer architecture, and to put advanced usage for operating system or compiler in their perspective.

Following is a list of policies for designing processor.

1) The specification is designed simple enough for first or second year students to learn essential of CPU operation.
2) The specification is designed in order to put advanced application for operating system or compiler in the perspective.
3) The specification that students have already learnt is adopted

In studies on CPU simulator, the items 1,2 are treated important factors. But our policy to decide the specification of processor is based on the item 1,3. This fact gives rise to a difference between ours and other specifications. That is, our specification is designed as a simple accumulator machine, while the specifications are based on register machines with load-store architecture. Since our idea is the correspondence between the interfaces of simulator and assignments, we can essentially apply the same idea to other CPU simulators for the introductory education of computer science.

The interfaces we designed are also used in CPU simulators. For example, the expression interface, E3 is same as the simulator for The Simple CPU [1]. The manipulation interface, M4 is same as one of interfaces of RTLsim [3]. One of differences between these simulators and our simulator is that these simulators use their own interfaces, while our simulator have a number of interfaces and use some of them along with assignments.

7. Conclusions

This paper describes a CPU simulator for introductory education of computer architecture. Our approach to design the simulator is based on the correspondence between the interfaces of our simulator and assignments of an exercise class on computer architecture. The simulator selects and provides a part of the interfaces for students along with an assignment. The simulator has single virtual CPU to simulate, and single processor block diagram to express the status of CPU. This framework helps teachers to design various assignments and exercises with CPU simulator on uniformed framework, and it also reduce students' extra learning overhead for CPU specification and use of the simulator.

We are applying the simulator to an exercise class at our Department of Takushoku University. We are going to evaluate the simulator with the practical use in the exercise class.

References

[1] J.R. Arias and D.F. Garcia, “Introducing computer architecture education in the first course of computer science career,” *IEEE Computer Society Technical Committee on Computer Architecture Newsletter*, July 1999, pp. 37-39.

[2] H.B. Diab and I. Demashkieh, “A computer-aided teaching package for microprocessor systems education,” IEEE Transaction on Education, vol.34, no.2, 1991.

[3] M. Pearson, D. Armstrong and T. McGregor, “Using Custom Hardware and Simulation to Support Computer Systems Teaching,” Proceedings of Workshop on Computer Architecture Education 2002, pp.19-26, 2002.

[4] K. Murakami and T. Ishikawa, "Introduction to Logic Circuit for Computer Systems" (Printed in Japanese), Kyoritsu Syuppan, pp.123-159, 1996.