16 Cards to Get Into Computer Organization

Siham Tabik and Luis F. Romero

Department of Computer Architecture
University of Málaga
29071 Málaga, Spain
E-mail: (felipe, stabik)@uma.es

Abstract. This paper presents a novel educative activity for teaching computer architecture fundamentals. This activity is actually a game that uses 16 cards and involves about twenty active participant students. Executing this activity in the first class of the course allows the student in only 45 minutes to acquire the fundamental concepts of computer organization. The results of the surveys that evaluate the proposed activity together with the grades obtained by the students at the end of course corroborate the importance of the proposed game in the assimilation of more complex concepts in computer architecture.

Keywords: Computer architecture basics, educative games, collaborative learning.

1 Introduction

The use of didactic games in higher education is a valuable educational process despite the reluctance of some professors who consider it a relaxation of the rigorous and solemn traditional methodology in university teaching. Indeed, Plato, the Greek philosopher and mathematician, considered the educational games an essential means for the formation of a perfect citizen and an excellent alternative to traditional methods, because they allow working with the skills of students by combining education, participation and fun.

According to Jean Piaget, the acquisition of new knowledge is built upon the previous knowledge and the learning process must be active in a way that the student receives from the professor only the tools to build knowledge but not the knowledge itself. These ideas acquire more relevance once extended to the collective learning in an interactive space [1, 2].

The combination of a structural and functional learning together with an analysis process in an educational game provides the essential requirements of a complete epistemic activity [3]. The pillars of the elaborated knowledge must be simple to guarantee that the student will focus on the essentials rather than having to worry about the details. Finally, the collaborative learning, in which students work together helping each other, guarantees the social dimension of the educational process [1–3]. These are the basics of the proposed game in which the students reproduce the functioning of the CPU.

On the other hand, the Computing Curricula 2001 [4] identifies 14 areas of basic knowledge in the discipline of Computer Science and each area specifies a set of units that comprises the core knowledge that a student should assimilate to obtain different types of degrees in Information Technology. The common requirements among several types of degrees is that the student must have a thorough knowledge of the fundamentals from the earliest computer courses.

The Instruction Set Architecture, i.e., the set of binary sequences that a computer is able to interpret as orders, has been traditionally identified as computer architecture. In fact, this instruction set is an abstraction of the considered hardware [6]. Therefore, understanding the concept of instruction and instruction cycle at ISA level is essential not only to understand the functioning of the computer but also to indirectly understand the foundations of the modern society.

There exist few works about active collaborative learning processes for novice CS students to help them understand the functioning of computers. For instance, Powers [8] proposed an active learning lab exercise, called "The living CPU", to help students to actively deduce the functioning of the CPU. The task of the teacher consists of providing the opportunity for this to occur. However, the content of this exercise is rather limited. Sherry [3] proposed an epistemic game based on a model processor to allow students to comprehend the role, function, structure and mechanism of a real computer. Nevertheless, the collaborative and collective aspect of this activity is limited. The activity proposed in this work not only covers the entire level of abstraction of the Instruction Set Architecture (ISA), which is of paramount importance for understanding the functioning of a processor, but also implies a high participation of both students and teacher.

This paper presents an educational activity in form of a role play in which the students represent the functioning of a computer without the need of previous knowledge. This game is specially appropriate for the first or second class of the subject of fundamentals of computers and takes about 45 minutes. The proposed activity consists of representing the functioning of a computer at ISA level via the execution of a code that occupies only 16 bytes in memory. The main contribution of this work is a careful selection of the content of the proposed code, where by means of only 14 instructions, 2 data in memory, and executing the mathematical factorial operation, it introduces all the essential concepts of the subject. The evaluation process showed very satisfactory results. In addition, this activity not only plays a crucial role to encourage student-teacher relationship, but also has shown to completely fulfill the drawn academic goals.

2 The 16 Card Game

The proposed activity is appropriate for the first or second class after giving a brief description of the history of computer science and after presenting a short biography of Von Neumann. Recall that the main objective of the proposed game is that the student comprehends the relevance of the architecture of Von Neumman. This section i) underlines the most important concepts for the un-

derstanding of Von Neumann architecture, ii) gives a description of the proposed 16 cards activity and its dynamics and, iii) provides the concepts introduced by each card.

2.1 Fundamental Concepts

The proposed activity aims i) to provide a suitable and complete introduction to the Instruction Set Architecture (ISA) and ii) to develop the knowledge of the student at the ISA level considering its duality hardware-software. We carefully selected a set of 15 specific concepts to substantially facilitate the learning process of the student [6, 7]. In particular, we identified:

- five concepts associated with the hardware aspect (Control Unit and DataPath (CU&DP), Input Output (I/O), General Purpose Registers (GPR), Arithmetic Logic Unit (ALU) and, STacK (STK)),
- five concepts associated with the software aspect (JuMP (JMP), CoNDitionals (CND), routines and functions (CLL), variables and Direct Memory Addressing (DAM) and, pointers (PTR)) and,
- five concepts associated with the functioning mode and the hardware/software interface (Instruction CYcle (ICY), Instruction SeQuencing (ISQ), Loads and Stores from memory (L/S), IMplicit Addressing (IMP) and , state of the computer (FLA)).

Introducing each one of these concepts is carried out using the every-day life analogies. For example, to understand the concept of pointers, the student finds in a box the number of the card that contains the data. As the game goes, all the concepts are revisited and the essential concepts are reinforced by adding details to ensure the progress of the learning process. Furthermore, during the course, as the concepts included in the agenda are appearing the professor will explicitly make reference to the activity proposed in this work. In this way, the spiral learning is going to be completed with an appropriate time spacing that favors the assimilation of the concepts in the long term memory [5].

Content of the 16 cards written in assembly language

- 0 LOAD R0,C
- 1 RESET R1
- 2 INCR R1
- 3 CALL A
- 4 MUL R1, R0
- 5 DECR R0
- 6 CMP R0,#0
- 7 BRNE 4
- 8 STORE D, R1
- 9 JUMP E
- A OUT [R1],R0
- B RET
- C –data– 4
- D –data– any
- E EXCH R0, R1
- F CALL A

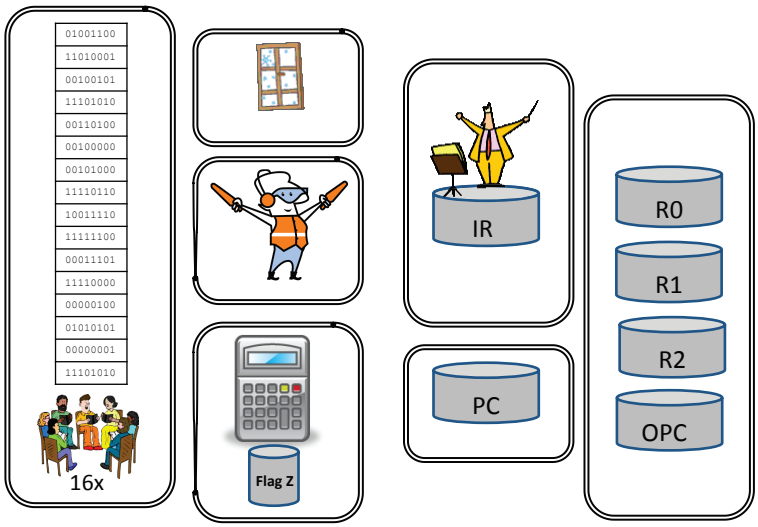

Fig. 1. The role of each player in the game.

2.2 The Cards

The professor prepares 16 cards printed in both sides. The first side, i.e., the side in green color, contains one of the instructions (or data) of the code shown in Figure 1. This side has a number that indicates the order of the instruction in the code written in base 10 system, i.e., from 0 to 15, and the instruction in plain English. In the back of each card, i.e., the side in orange color, there is a double translation of the content of the card in assembly and binary programming languages. The professor provides an additional dictionary material that provides the translation of all the used instructions from binary to assembly language, e.g., RESET=0010, C=1100, so that the student can understand that the content of the back of the card is in fact the translation of what the computer stores in memory and registers. Only the first side of the cards is used in this activity, and the additional information in the back will be revisited in next classes.

Fourteen cards contain instructions and two cards contain data of type integer, one of them is modified during the game to store the partial result of the factorial.

2.3 The Players

The professor selects a group of students, generally 22 students, and distributes the previously prepared 16 cards among them. The rest of the students are assigned a role each one. The professor plays the role of Control Unit that governs the computer and the students are assigned the following tasks:

- 16 students represent 16 memory positions. Each one is assigned one of the 16 cards. A random distribution of the cards will help later to understand the concept of Random Access Memory (RAM)
- 1 student represents the ALU unit. He is assigned a calculator and a box labeled Flag Z.
- 1 student represents the bank of registers (BR). He is assigned 4 boxes labeled R0, R1, R2 and OPC.
- 1 student represents the memory controller (MC).
- 1 student plays the role of the I/O unit. This role is usually assigned to a student close to a door or a window of the class.
- 1 student represents the program counter (PC).
- 1 student or the instructor plays the role of instruction register (IR)

Figure 1 sketches the role of each player.

2.4 Dynamics of the Game

The professor informs student PC that the game starts. This student tells his content which is always a number initially equals 0. Student MC searches for the student whose card contains this number and asks him to tell his content loudly. This content has two orders. The first order is a common sentence to all the memory cards, which increments PC. The second order is a simple instruction written in plain English. The professor asks PC to sum one to the value he memorizes and write down in the board the mnemonic of the second instruction of the card.

After fetching the first instruction, the professor indicates that the process is mechanical and normally it is performed in picoseconds. The professor also points out that in practice the used language is the binary and the increment of PC is not explicitly written in the machine instructions. Computer architectures have implicit sequencing.

After the search operation is completed, the professor executes the instruction by selecting the student that has been assigned the corresponding role and asking him for example: "MC, please, could you localize card C" then, "card C, could you please tell us your content", "BR, could you please store the value in register R0".

The game keeps executing all the instructions till PC reaches the value 16. In summary, 28 instructions have to be executed including 30 memory accesses, two writes in the ports (the first writes the value to which the factorial is calculated and the second writes the result of the operation). Figure 2 shows the flow of the instructions of the code.

As the game advances, the professor can make jokes, for example, about the cards executed out of order like card 4 and A. After the execution of card 3, the professor can tell the student with Card 4:"Are you ready? You are going to work more then anyone else". Normally from the second iteration of the loop, the professor stops again to indicate that the processor is substantially much faster and that he should do the same.

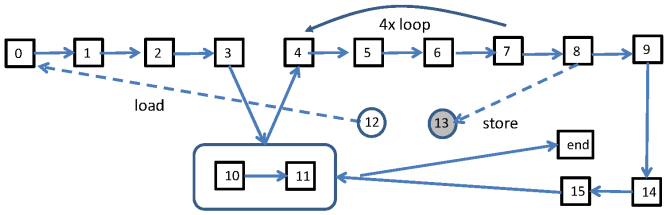

Fig. 2. Sequence of execution of the cards.

At the end of this activity, the professor stresses that the tasks performed by each participant are in fact very basic tasks, however, thanks to the high number of resources or transistors and the high rhythm of work, on the order of Gigahertz, we can perform relatively complex tasks, such as the calculation of the factorial of a number, in a small interval of time.

2.5 Relationship Concepts-Cards

The content of the cards and its relationship with the 15 fundamental concepts fixed previously is as follows:

- **Card 0** (LOAD R0, C): This instruction copies the content of Card C into Box R0. It is used to introduce two fundamental concepts in computer structure, the Central Processing Unit (CPU) and the buses that communicate different memories, in addition to three concepts from computer architecture, instruction cycle, instruction sequencing and memory cycle. It is recommended that the professor spends five minutes in this first card since it will serve simultaneously to introduce the pillars of the subject.
- **Card C** (–data– 4): This read-only data address has two objectives: i) to reinforce the memory cycle and differentiate it from the instruction cycle and ii) to explain the idea of the unique memory of instructions and data in Von Neumann architecture. The professor points out at this stage that a well designed program should find data and instructions where they are supposed to be.
- **Card 1** (RESET R1): Set R1 to 0. This card reinforces the concept of implicit sequencing through the program counter and the instruction cycle. It is also used to explain the concept of equal sign, e.g., A=B, in programming languages. That is, what the left hand side term stores after the execution of the instruction is what was calculated in the right hand side term during the instruction.
- **Card 2** (INCR R1): This card is used to introduce very different concepts although it is very similar to the previous one. The first concept to be pointed

out is that from the computer structure point of view, there is a big distinction between CU (represented by the professor) and the ALU. The second concept consists of introducing the idea of the code of operation, addressing by register and implicit addressing. In addition, the implicit increment of PC should be recalled at this stage too.

- **Card 3** (CALL 10): This instruction copies PC in OPC then sets PC to 10. This in probably one of the most important cards. It introduces the explicit sequencing and the imperative programming. The students with some previous basic knowledge in high level programming languages are able to identify the call to routine as low level operations. This card is also used to indicate the reason behind why the implicit increment of PC is performed at the beginning of the instruction cycle.
- **Card A**(OUT [R1], R0): This instruction consists of showing the content of box R0 at the port indicated by box R1. This card introduces three new concepts:
  - indirect addressing, i.e., pointers.
  - Input/output interface, which communicates the information to the exterior through one of the windows of the classroom.
  - General purpose registers
- **Card B** (RET): Return instruction indicates the end of the routine. It complements the description of the procedural programming. In addition, if considering a recursive call, this allows introducing the idea of substituting OPC by a Stack of registers.
- **Cards 4 and 5** (MULT R1, R0 and DECR R0): The first instruction multiplies box R0 by Box R1 and stores the result in box R1. The second instruction decrements the content of box R0. Student ALU is asked to execute these two instructions. At this stage the separation between control unit and data path is emphasized.
- **Card 6** (CMP R0, #0): Does box R0 has the value 0? If this is the case, store the value 1 in box Flag Z. This instruction introduces the concept of the flag and state register. The first execution of this sentence isolates form the next bifurcation to increase the student intrigue.
- **Card 7** (BRNE 4): Branch to 4 if flag Z is 0. This is also one of the most important instructions, it allows the student to understand that the program can change its execution in terms of the data.
- **Card 8** (STORE D, R0): This instruction stores in position D in memory the content of box R0. It introduces the operation of writing in memory and reinforces the concept of Von Neumann architecture.
- **Card D** (–data– any): The value of box R0, which should be 24, is written in this card. This is the only card that will be modified. This aspect is emphasized due to its relevance in the design of the memories. The concepts ROM and RAM are also introduced.
- **Card 9** (JUMP E): Using this card allows the student to distinguish between 4 sequencing modes in a computer, implicit, procedural, branch and explicit jump sequencing.

- **Card E** (EXCH R0, R1): This instruction switches the contents of R0 and R1. Together with the previous card, this card reinforces the concept ISQ. This instruction *per se* is not didactically relevant although it is used in class to talk for the first time about computers with complex instructions (CISC) versus computers with simple instructions (RISC).
- **Card F** (CALL 10): The execution of this instruction copies PC in OPC, then sets PC to 10. This instruction is used to reinforce the concepts of imperative programming. The second call to the same function allows the student to check that the use of functions reduces the size of the code.
- **Cards A and B** ($print(R0, R1)$ and RET): Already explained. The multiple execution of these function reinforces the concept of procedures and the use of pointers.

Table 1. The concepts attributed to each card. The symbols -, + and, ++ indicate an increasing intensity of emphasis on each concept.

Card	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ICY	++	++	+	++	-	-	-	-	-	-	-	-	+	+	-	-
ISQ	++	++		++	+			++		++	+	++			++	++
JMP				++				++		++		++			+	++
FLA							++	++								
CND							+	++								
CLL				++				+			+	+			+	++
STK				+								++				+
CU&DP	+	++	+		++	++			+							
ALU		+	++		++	++		+							++	
L/S	++								++		+		++	++		
GPR	+	+	++		-	-	-		+		++				-	
I/O											++					
PTR										+	++					
DAM				+					++	++	+		+	+		+
IMP		+	++		+	++	-					++				

Table 1 summarizes the concepts associated to each card indicating the intensity of emphasis on each one. As some cards are reused many times during the game, the intensity may vary slightly either in an ascending or descending way.

3 Evaluation of the activity

Although the practice of this activity started in 1997/98 with the students of the third course of industrial engineering, subject Computer Fundamentals, it was not evaluated till the academic year 2000/2001. Since then, 538 surveys were carried out with the objective of determining the incidence of the activity on the

Table 2. Level of knowledge of the concepts and degree of influence of the activity on the learning of each one of the concepts in the course 2011-2012

	Level of knowledge			Influence of the activity		
Concept	stage 1	stage 2	stage 3	stage 1	stage 2	stage 3
ICY	5.0	4.1	5.0	4.2	4.2	**4.3**
ISQ	4.8	4.0	4.6	4.3	4.3	**4.1**
JMP	4.8	3.3	4.9	4.5	4.5	**4.0**
FLA	3.5	2.6	4.4	5.0	5.0	2.2
CND	4.2	2.9	4.7	4.0	4.2	2.7
CLL	4.0	3.8	4.6	4.0	4.2	**3.4**
STK	3.8	2.1	4.3	4.4	4.6	1.9
CU&DP	4.6	3.5	4.9	4.9	4.9	**3.2**
ALU	4.8	4.3	4.9	4.3	4.0	2.3
L/S	3.7	2.5	4.8	4.6	4.7	1.5
GPR	4.4	4.3	4.9	4.8	4.9	1.0
I/O	4.9	4.7	4.9	5.0	5.0	**3.2**
PTR	3.2	1.5	4.1	4.8	4.8	0.8
DAM	3.3	2.0	4.7	5.0	5.0	1.1
IMP	4.6	3.8	4.8	5.0	4.9	**3.0**

learning process of the subject. The survey was partitioned into three stages. The first evaluation is performed right after the activity in the first class, the second evaluation is performed two weeks after the week of the activity and the third evaluation is done at the end of the course.

The survey asks the students two questions i) to evaluate their own level of knowledge in each one of the 15 concepts described in Section 2.1 using marks between 0 and 5 and also ii) to quantify the degree of influence of the activity in the assimilation of the concepts, an evaluation equals 0 means the activity did not have any influence in the knowledge of the concept while 5 means that the student learnt the concept only through the activity.

In general, the results, appropriately filtered[1], are very satisfactory since in the first stage the students think that they understood well the concepts, see Table 2. Moreover, two weeks later the results of the activity which has not been mentioned yet in class are still present in the memory of the students. At the end of the course, the students confirm that they knew 7 concepts (ICY, ISQ, JMP, CLL, CU&DP, I/O and, IMP) mainly through the activity presented in this work. For the rest of the concepts, the students think that about 20% of their knowledge is also thanks to the activity. The only concept that has not been affected by the activity proposed in this work is the pointers due to the fact that another specific activity were especially designed for this purpose.

The grades obtained by the students along the course could have been considered in the evaluation of the influence of the activity in the learning of the

[1] Discarding invalid and malicious responses

fundamental concepts. However, we should not consider it because the improvement of the average grade also depends on other factors such as the experience of the professor, the increase of the average mark for accessing the degree and, a better preparation of the students in the field of information and communication technologies.

On the other hand, for example in the course 2012-2013, the professor who implemented this activity got a very good evaluation by the students, 4.79/5 in 38 independent surveys carried out by *Centro Andaluz de Prospectiva* [10]. This evaluation exactly equals the mark that the student attributed to this specific question: "The professor uses didactic resources to facilitate the learning?". It is interesting to indicate that the average evaluation attributed by the students to the professors in the University of Málaga, is 4.70. This data indirectly provides positive additional evaluation about the relevance of this activity in teaching fundamentals of computers.

4 Conclusions

This work presented an activity that substantially easies the assimilation of the fundamental concepts in the functioning of a computer. The contents have been carefully elaborated to maintain simplicity. The results of several surveys demonstrated that the proposed activity is extremely useful to reach the drawn objectives.

References

1. Moreno L., Gonzalez C., Castilla E., Gonzalez E., Sigut J., (2007). Applying a constructivist and collaborative methodological approach in engineering education. Computers & Education, 49(3):891–915.
2. Ben-Ari M., (2001). Constructivism in Computer Science Education, Journal of Computers in Mathematics and Science Teaching, 20(1):45–73.
3. Sherry, L., (1995). A model computer simulation as an epistemic game, SIGCSE Bull, 27(2):59–64.
4. ACM, IEEE Computer Society (2001). Computing Curricula 2001. The Overview Report, Los Alamos, CA, The Join Task Computing Curricula.
5. Bruner J., (1960). The Process of Education, Massachusetts: Harvard University Press.
6. Hennesey J.L., Patterson D.A., (2012). Computer architecture, a quantitative approach, Massachusetts: Morgan Kaufmann.
7. Patterson D.A., Hennesey J.L., (2009). Computer Organization and Design: The Hardware/ Software Interface, Massachusetts: Morgan Kaufmann.
8. Powers K.D., (2004). Teaching Computer Architecture in Introductory Computing: Why? And How?, Conferences in Research and Practice in Information Technology, 30: 255-260.
9. Stanley T.D., Wang M., (2005). An emulated computer with assembler for teaching undergraduate computer architecture, Proceedings of the 2005 Workshop on Computer architecture education, 7.
10. Centro Andaluz de Prospectiva. http://canp.es/.