Approaches and Tools used to Teach the Computer Input/Output Subsystem: A Survey

Edurne Larraza-Mendiluze and Nestor Garay-Vitoria

Abstract—This paper surveys how the computer Input/Output (I/O) subsystem is taught in introductory undergraduate courses. It is important to study the educational process of the computer I/O subsystem because in the curricula recommendations it is considered a core topic in the area of knowledge of computer architecture and organization (CAO). It is also a basic knowledge to be acquired in order to work in areas such as human computer interaction (HCI) or embedded systems.

Examination questions, course syllabi and textbooks were analyzed to identify which teaching approaches are being used. Individuals teaching the I/O subsystem could choose between the options explained here, according to their intended learning outcomes.

In addition, a literature survey was conducted on the development and use of tools to improve student understanding of I/O, and to make the topic less abstract and more attractive. A goal is to indicate to computing education researchers that the majority of the literature report experiences in developing or using different resources or educational methodologies, but that these are not based on a theory of learning.

Index Terms—Computer architecture, computer input/output subsystem, curricula, education, survey.

I. INTRODUCTION

A computer can serve no useful purpose unless it communicates with the external environment. Instructions and data stored in memory must come from some input device. Computational results must be transmitted to the user through some output device” [1]. These external devices, together with controllers, channels and Input/Output (I/O) processors, form the computer I/O subsystem. In other words, “the computer I/O subsystem is the interface between the computer and the outside world” [2]. “The I/O subsystem architecture deals with the organization of concurrent processing activities within all the components of the I/O subsystem” [3].

The computer I/O subsystem is considered by the ACM/IEEE-CS Joint Task Force as a core topic in both their computer science [4] and computer engineering [5] curricula recommendations. These curricula recommendations use the term “Interfacing and communication” to refer to the topic. There is a major difference between the two curricula: the CS 2013 curriculum [4] focuses on the knowledge level (with a minimum coverage time of two hours); the CE 2004 curriculum [5] demands as learning outcomes the ability to write small Interrupt Service Routines (ISRs) and I/O drivers, and the ability to analyze and implement interfaces (with a minimum coverage time of ten hours).

The I/O subsystem is also an important part of the human computer interaction (HCI) subject area. When designing interfaces based on non-standard devices (i.e., not the classical keyboard, screen and mouse), developers have to know a diversity of I/O devices and their characteristics [6].

In addition, the use of interrupts to synchronize the I/O subsystem with the central processing unit (CPU) introduces the students to concepts such as multiprogramming and concurrency, which offer increased system performance to that achieved with mono-programming and synchronization via polling.

The I/O subsystem topic introduces these concepts and is often considered as being at the introductory level [7], [8]. However, there are at least two problems with the educational process for this topic.

The first is the level of abstraction with which students are faced. It is very easy to see how a mechanical typewriter converts a key press into a letter on the paper. In contrast, most computer users will be unable to identify the actions that take place between their pressing a key on a computer keyboard and the corresponding letter appearing on the computer screen.

The second problem is the level of confidence of the people teaching in this subject area. In a survey administered to 82 individuals teaching in the CAO area, the authors [9] found that “fault handling & reliability and synchronization & handshaking are apparently poorly understood, but are taught by only a small percentage of instructors. [...] the sub-areas of bus systems & direct memory access (DMA), I/O control methods, interrupts, and external storage [...] are taught by about one-half of respondents and the low self-confidence index of a large portion of them indicates this is an area where instructors could use help”.

This up-to-date literature review for the computer I/O subsystem as a teaching topic will be of value both to those new to teaching this topic and to those who have been teaching the topic for some time. The paper presents various teaching approaches and links these to their outcomes.

The technology available for teaching is also discussed. This paper is also aimed at computing education researchers, in that the majority of the studies found in the literature report experiences of using different resources or educational methodologies, but are not based on a theory of learning.

Section II of this paper shows the steps taken to identify the...
various approaches used to teach the computer I/O and then
describes them. Section III reviews the literature and presents
the tools used to teach the computer I/O subsystem. Finally
Section IV summarizes the conclusions and proposes some
future work.

II. VARIOUS APPROACHES USED TO TEACH THE
COMPUTER I/O SUBSYSTEM

A. Data Gathering and Analysis Process

Three studies were conducted as part of defining the classi-
fication of the different learning approaches for the computer
I/O subsystem.

The first study was on exam questions. The authors of
this work analyzed six different Spanish universities’ exam
questions testing their students’ knowledge of the computer
I/O subsystem [10]. This analysis showed that pure or applied
knowledge recall was needed to answer 30% of the exam
questions, some coding skills were needed to answer 30% of
the questions, and analysis of performance was required in
40% of the questions. This provided an initial classification of
the learning approaches; the other two studies were used to
complete the classification.

The aim of the second study was to analyze various uni-
dersities internationally. One of the problems was to select the
universities to be analyzed. Six well known university rankings
[11]-[16] were used for this purpose. The 36 universities
ranked among the top 100 in all the ratings were selected and
their curricula were reviewed. The review analyzed the
syllabi, assignments and exams of undergraduate courses that
included the computer I/O subsystem. In eight cases, no
information was found or the information was inadequate.
In nine other universities the I/O subsystem was found to
be taught in operating systems courses or from an operating
systems point of view and these were therefore not considered
for this study, which has a clear CAO perspective. As a first
approach, the syllabi, assignments and exams available online
from the remaining 19 universities were analyzed. Some of
these universities had no exercises devoted to the computer
I/O subsystem in their exercise list or in their exams. In the
Computation Structures course at the Massachusetts Institute
of Technology (MIT) tutorial exercises ask questions on topics
such as the time devoted to servicing and interrupt or the
running time of a program when some specific interrupts are
enabled. Finally, some other universities have labs or
projects in which students have to program ISRs (sometimes in
assembler and sometimes in higher level languages) in order,
for example, to print a string on the screen, program an alarm
clock with Arduino, or program a game for the Game Boy
Advance (GBA).

Finally the third study considered several widely-used
textbooks. Textbooks listed in [9] as being widely-used in
computer architecture and organization courses [1], [2], [17]-
[20], were selected as a starting point for this study. The
latest editions of these textbooks were selected. [1] and [19]
were last published in 1993 and 1992 respectively, so were
initially considered to be too old and discarded. However, an
Internet search revealed that [1] is still widely used, so it was
therefore reconsidered. Since [18] does not say much about
the computer I/O subsystem, it was also discarded. Tanenbaum
and Austin mainly describe how certain peripherals are built,
then they mention the I/O techniques, and finally they talk
about I/O in different operating systems. The textbook [17]
poses questions and problems on the performance of the
system. Reference [20] treats embedded systems directly and
the I/O subsystem is considered from a programming point of
view. Finally, [1] focuses on the path of the data going into
and out of the computer and the signals used to synchronize
devices, CPU, and memory, including a lot of block and
 timing diagrams and flowcharts. However, in this book there is
also some low-level programming of the I/O subsystem using
assembly language.

B. Learning Approaches

The following teaching approaches and their characteristics
were extracted from the three studies described above:

a) The purely descriptive approach: In this approach
only a description of the computer I/O subsystem is given to
the students, who are then expected to be able to describe
the concepts. At most, they could be asked to identify relationships
between concepts. This is an easy way to introduce the topic,
that could even be used with students who are not majoring in
computing. In knowledge taxonomies, such as those presented
in [21] or [22], this approach would be classified as low level.

b) The performance approach: The computer I/O sub-
system is a bottleneck in the computer system. Its design
has a major effect on the computer’s performance. In this
approach, the students are asked to calculate the performance
of a computer system using different I/O techniques. This
kind of question is considered a higher-level question, since it
requires the application of knowledge and often the evaluation
of the results to determine which technique is most appropriate
for the context [21].

Performance is a key feature in computing, and students
must be made aware of this. However, in the I/O subsystem
the final element is often a human being; when that happens it
is difficult to justify the use of a particular I/O option. In such
cases the bottleneck may not be improved from the computing
perspective.

c) The programming approach: Given the recent trend
in computing towards embedded systems and to the increasing
attention being paid to HCI (where the I/O subsystem is
crucial [6]), and considering the last Computer Engineering
curriculum of the ACM/IEEE-CS Joint Task Force [5], it is
also important to see the functioning of the computer I/O
subsystem from a programming point of view (accessing the
I/O registers, programming the ISRs, etc.).

This approach helps to introduce very abstract concepts such
as interrupts or concurrency. Programming the computer I/O
subsystem can help to make these concepts clear. However, the
problem is that in undergraduate introductory courses students’
programming level may not be sufficiently high, or if it is, that
low-level programming will not be very appealing to them.

d) The data-path signal approach: Although the last
edition of [1] is from 1993, it was found to be the fourth most
widely used textbook in teaching computer organization and architecture in [9], and a quick search on the Internet showed that it is still being used in several courses. This book presents a different approach which can be very helpful for envisaging how the hardware works. However, it remains at the level of theoretical understanding, rather than practical experience, since getting students to make changes in the hardware can be really difficult. This could be done at a more advanced level.

Tables I and II classify the universities and textbooks under these four approaches, according to the criteria shown in Section II-A.

III. TOOLS USED TO TEACH THE COMPUTER I/O SUBSYSTEM

This section presents an up-to-date review of the tools being used to teach the computer I/O subsystem in introductory-level courses, not to make a comparison between tools. The information used is based solely on material presented in the literature on the use of these tools and only if the tools are still currently available. Previous surveys, such as [23] or [24], were not considered because they did not report the tools used to teach the computer I/O subsystem. Work prior to the year 2000 was not considered either; educational tools must be attractive to students and in that context older technology cannot compete.

Software tools such as simulators, emulators or virtual machines have been widely used for teaching CAO because “they allow students to gain valuable hands-on experience in design and programming, without the problems involved in maintaining a hardware lab” [25] and because they “give students a better appreciation for the complexity of their own computers” [26].

Despite the extensive use of software tools, some authors argue that “students must touch, feel and smell the real hardware in a computer organization course” [27] and that the use of software tools “fails on concreting theoretical concepts into real ones” [28].

The tools presented below are classified according to the aforementioned approaches, with the reported results of using of these tools being provided for the benefit of computing education researchers.

- Scott [29] reports the virtualization of the computer presented in [1], with a graphical user interface (GUI) and a built in assembler. The tool can be found in [30]. In this virtual computer a poll or an ISR can be programmed in a fictitious assembly language, and the interface shows the changing values of the registers and memory as the instructions are executed. Due to the simplicity of the Mano computer, only one character can be input or output at a time. Moreover, in the example shown the ISR and the interrupt controller tasks appear mixed. This is a tool that can be classified as belonging not only to the programming approach but also to the data-path signal approach, because its GUI offers the possibility of changing flags and seeing the changes in registers and memory.

This paper does not report usage results.

- Far newer is [25], in which Donaldson et al. argue that many tools, like the one presented above, tend to focus on only one aspect of the course. In order to solve this problem, they present a GUI-based digital logic simulator (DLSim3) with a collection of circuits and plug-ins (DLsys) covering all aspects of the computer architecture. The platform can be found in [31], and contains several plug-ins that simulate I/O devices. They also suggest some exercises. As for the I/O subsystem, they suggest connecting several I/O devices to a single interrupt request line using a daisy chain or modifying the CPU plug-in so as to separate the I/O address space from memory address space. Since the tool offers the possibility of changing the system itself it can be considered as a representative of the data-path signal approach.

The system had not been tried out with students at the time of publication and the authors of this paper could not find any later data.

- Black and Komala [26] report the building of a full system emulator after finding that in existing simulators “peripherals and I/O are generally neglected”. The tool is written in Java and can be downloaded as a single JAR file or run from a web-page as an applet. For the I/O subsystem they developed a lab where “students write a simple graphical animation program (a moving car) in x86 assembly. This involves configuring the interval timer, modifying the interrupt vector table and making their own timer interrupt routine, switching the video to graphics mode, and writing to video memory. Because the simulator accurately models a PC the students can then run the same program on their own laptops”. They can also control hardware connected to the serial port. Black and Waggoner [32] report the integration of the x86 full system simulator with a simulator to teach CPU design. The integrated system contains a data-path builder, a control builder and a processor wizard. Considering the lab proposed in [26], this tool could belong to the programming teaching approach. However, in [32] the authors require the students to design their own processor, considering all the data-paths. This tool could therefore also be considered to belong to the data-path signal teaching approach. The authors of [32] report usage experience from the student point of view, but they do not report learning benefits.

- Ellard et al. [33] present the Ant-32 architecture. This is a 32-bit RISC architecture designed specifically for educational purposes. A tutorial for the Ant-32 assembly language can be found in [34]. They do not use a real architecture, arguing that these are too complicated and require mastery of too many arcane details to accomplish anything worthwhile. The implementation of the architecture they present uses a simple (but full-featured) bus architecture that was originally designed for use with the Microprocessor without Interlocked Pipeline Stages (MIPS) processor architecture that allows the use of simulators for devices already written for that bus and has special treatment for interrupts and exceptions. The tool could be defined as belonging to both the programming and the data-path signal teaching approaches. Since no exercises are mentioned, this cannot be clearly determined.

No usage data of any kind is reported in [33].

- Brorsson [27] presents the MipsIt system, which consists of a development environment, a hardware platform and a series of simulators, aimed at the Windows (95-XP) platform.
as host machine. The evaluation board (containing an IDT 36100 micro controller with a MIPS32 ISA processor core) and the simulators are carefully explained in the paper. A simulator of the whole system can be found in [35], and the lab exercises proposed for the I/O subsystem topic can be found in [36]. For the I/O subsystem, Brorsson developed “a simple daughter board containing one 8-bit and one 16-bit parallel bi-directional I/O port”, which “also contains a simple interrupt unit with three interrupt sources (two push-buttons and one adjustable pulse source) that could also be read as a 6-bit parallel input port”. Lab exercises first require the students to analyze some code to understand the differences between polling and interrupt-driven I/O. Then they have to execute the code to observe what happens, and finally they make various changes to obtain further results. This tool could clearly help in a programming teaching approach.

Once again, no data about the usage of the tool was reported.

- In Teller et al. [37], “after acquiring some competence at programming the 68HC11”, using the Visual 6811 simulator (a simulator for the Motorola HC6811 microprocessor), “students are challenged to program small robots that are controlled by 68HC11s”. “The first robot lab has students use the serial communication interface to display the contents of the robot’s memory (the 68HC11) on the monitor of the host PC. Pressing a key on the host’s keyboard, which instructs the 68HC11 to read from or write to a memory byte or word, drives the related program […] The first final project had students program robots to navigate a maze. […] Another challenging final project was to program two robots equipped with IR detectors: a wimp and a follower”. This tool can clearly be classified in the programming approach.

In [37] the authors assessed the use of the tools via questionnaires, where they discovered that the majority of the students thought it was a good idea to use the robots on the course and that programming the robots helped to reinforce the concepts they had learned and helped them to see (physically) the power of the 68HC11 architecture in action.

- Brylow and Ramamurthy [7] report the use of the Linksys WRT54GL family of wireless routers in two different courses. In this paper the focus will be on the sophomore level course at Marquette University. The system “contains a little-endian embedded MIPS 32 processor […] with 16 MB of RAM, and 4MB FlashROM. […] easily accessible serial port connections on the main board that allow direct access to the device firmware”. Of the exercises they include, the two most relevant to the I/O subsystem state as follows: “Rather than view a device driver through the lens of a heavily abstracted operating system layer, students work directly with the memory-mapped control and status registers of a serial device to build input and output primitives that they will reuse for the remaining of the term”, and “Working directly with I/O interrupts, hardware FIFOs and multiple devices, students confront the genuine complexity of interaction between embedded processors and their peripherals”. This is a tool to be classified in the programming teaching approach.

This study again reports students’ satisfaction, but contains nothing on learning outcomes. More information on this study can be found in [38].

- Santofimia and Moya [28] report changing the PIC16F84 for a Nintendo DS (NDS) gaming console. In this case, the interrupt concept is introduced via timers and scrolls. “Implementation of counters can be accomplished by using timers”. Moreover, interrupts generated when line drawing are necessary to display new data on the screen. Here again there is a tool devoted to providing help in the programming approach.

Students were given the option of choosing between PIC16F84 and NDS. “Over thirty percent of the students chose the NDS platform, [which] provided an acceptable feedback

<table>
<thead>
<tr>
<th>University</th>
<th>Subject</th>
<th>Approach</th>
<th>Book</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cornell University</td>
<td>CS 3410: Computer System Organization &amp; Programming</td>
<td>Descriptive</td>
<td>PatHen [17]</td>
</tr>
<tr>
<td>Duke University</td>
<td>CS 230: Computer Architecture</td>
<td>Descriptive</td>
<td>PatHen [17]</td>
</tr>
<tr>
<td>National University of Singapore</td>
<td>CS 2100: Computer Organization</td>
<td>Descriptive</td>
<td>PatHen [17]</td>
</tr>
<tr>
<td>New York University</td>
<td>CSCE 0436: Computer Architecture</td>
<td>Descriptive</td>
<td>PatHen [17]</td>
</tr>
<tr>
<td>Northwestern University</td>
<td>EECS 361: Computer Architecture</td>
<td>Descriptive</td>
<td>PatHen [17]</td>
</tr>
<tr>
<td>Swiss Federal Institute of Technology Zurich</td>
<td>Systems programming and computer architecture</td>
<td>Descriptive</td>
<td>PatHen [17]</td>
</tr>
<tr>
<td>University of California San Diego</td>
<td>CSE 111: Computer Architecture</td>
<td>Descriptive</td>
<td>PatHen [17]</td>
</tr>
<tr>
<td>University of Edinburgh</td>
<td>INPC: Computer Systems</td>
<td>Descriptive</td>
<td>PatHen [17]</td>
</tr>
<tr>
<td>University of Michigan</td>
<td>EECS 370: Introduction to computer organization</td>
<td>Descriptive</td>
<td>Others</td>
</tr>
<tr>
<td>University of Texas at Austin</td>
<td>CS 352: Computer Systems Architecture</td>
<td>Descriptive</td>
<td>PatHen [17]</td>
</tr>
<tr>
<td>University of Washington</td>
<td>CSE 352: Hardware Design and Implementation</td>
<td>Descriptive</td>
<td>Others</td>
</tr>
<tr>
<td>Massachusetts Institute of Technology (MIT)</td>
<td>Computation Structures</td>
<td>Performance &amp; Programming No textbook</td>
<td></td>
</tr>
<tr>
<td>California Institute of Technology (Caltech)</td>
<td>EECS 51: Principles of Microprocessor Systems</td>
<td>Programming No textbook</td>
<td></td>
</tr>
<tr>
<td>Georgia Institute of Technology</td>
<td>CS 2110: Computer Organization and Programming</td>
<td>Programming Not found</td>
<td></td>
</tr>
<tr>
<td>McGill University</td>
<td>COMP 273: Introduction to Computer Systems</td>
<td>Programming</td>
<td>PatHen [17]</td>
</tr>
<tr>
<td>Purdue University</td>
<td>CS 250: Computer Architecture</td>
<td>Programming</td>
<td>No textbook</td>
</tr>
<tr>
<td>University of Amsterdam</td>
<td>Architect &amp; Computerorganisatie</td>
<td>Programming</td>
<td>Others</td>
</tr>
<tr>
<td>University of North Carolina Chapel Hill</td>
<td>COMP 541: Digital Logic and Computer Design</td>
<td>Programming</td>
<td>Others</td>
</tr>
<tr>
<td>University of British Columbia</td>
<td>CPSC 213: Introduction to Computer Systems</td>
<td>Programming</td>
<td>Others</td>
</tr>
</tbody>
</table>

TABLE I
UNIVERSITIES, COURSES, THE APPROACH THEY FOLLOW, AND TEXTBOOK/S USED

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Purely descriptive</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Performance</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Programming</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Data-path signal</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

TABLE II
APPROACH FOLLOWED BY EACH TEXTBOOK
for comparing results on the acquired knowledge.” Using the NDS the results showed “an important improvement in terms of the contents understood and retained by students.”

• Larraza-Mendiluz et al. [39] also use NDS in their approach, in this case switching from a PC-based laboratory to an NDS-based laboratory. The paper has a section that “shows the extent to which the NDS I/O subsystem was used in [the] study and explains the various elements involved in the I/O process”. The tool can be used to control peripherals by programming the polling or the ISRs, and changing the registers of the devices. Therefore, this fails under the programming approach.

The paper presents data gathered in two consecutive years, where the experimental group used the NDS in both years and the control group used the PC during the first year and NDS during the second year. While in the first year the scores obtained in the control group were lower than those in the experimental group, for the year in which both groups used the NDS the scores were almost the same. It is noteworthy that the experimental group also had lower drop-out ratios.

IV. CONCLUSIONS AND FUTURE WORK

This paper has presented a survey related to the educational process for the computer I/O subsystem at the undergraduate introductory level. The survey of textbooks, course syllabi, and conference and journal papers has led to the authors defining a classification of different teaching approaches. Advantages and disadvantages of these learning approaches have been suggested. Moreover, several tools found in papers have been selected based on their suitability for teaching the I/O subsystem. Without obvious metrics for evaluating how different approaches/tools affect the knowledge gained by students, it is hard to rank these methods according to a clear outcome. However, this paper provides a good overview of the outcomes intended with each approach and tool.

This work also highlights the need for more educational research in the I/O subsystem topic. Many tools have been developed, but most have not been evaluated. Nor are there clear outcome objectives in what is an important field, not only for CAO but also for embedded systems and HCI.

The survey was limited to introductory courses taught at the undergraduate level. As future work it would be interesting to analyze how the educational process for the computer I/O subsystem is evolving in more advanced courses.

REFERENCES


Edurne Larrazar-Mendiluze received a M.S. degree in computer engineering from the UPV/EHU in 1999, a M.S. in ICT applied to education from Mondragon Unibertsitatea (MU), in 2004, and a Ph.D. in computer science from the UPV/EHU in 2014. She is an Assistant Professor in the Computer Architecture and Technology Department of the UPV/EHU. Her research interests include computing education.

Nestor Garay-Vitoria has a M.S. degree in informatics and a Ph.D. in computer science from the UPV/EHU in 1990 and 2000, respectively. His Ph.D. thesis received the extraordinary Award in October 2002. He is an Associate Professor in the Computer Architecture and Technology Department of the UPV/EHU. He was the first Academic Director of the Master's program on Assistive Technology for Personal Autonomy given in UPV/EHU. His research interests include human-computer interaction for special needs, in Egokituz, a high performance research group for the Basque Government.