Development of a Virtual Computer Architecture Course

Silvia Bechter, Torsten Braun, Günther Stattenberger
Institut für Informatik und Angewandte Mathematik, Universität Bern
Email: [bechter|braun|stattenb]@iam.unibe.ch, URL: www.iam.unibe.ch/~rvs

Keywords: Distance Learning, Computer Architecture, Practical Exercises, Simulation

1. INTRODUCTION

Virtual exercises and lectures are being developed to assist or replace many traditional lectures. A lot of them are rather straight-forward, e.g. develop multimedia content, or add on-line tests and quizzes to a web page of a lecture. However, there are several types of courses it is rather difficult to develop a virtual course for. Examples are practical exercises, where students have to work with special equipment within a laboratory room. Several efforts to replace laboratory exercises by virtual ones have been started. For example, within the Swiss Virtual Campus programs, the projects Nanoworld [5] and Vitels [6] are developing courses allowing students to perform practical exercises virtually and / or remotely from any workstation. In both projects, significant technical efforts are required to build a secure and reliable infrastructure supporting users in performing virtual and remote experiments. In this paper we describe another virtual exercise. In contrast to the courses mentioned above, no externally funded project has been set up for its development. Moreover, we did not have the explicit goal to develop a virtual course. The virtual course just evolved by changes to a traditional course that have been necessary due to several financial and organisational constraints. The result of the different actions finally was a course that can completely be performed remotely using a web-browser.

2. TRADITIONAL COMPUTER ARCHITECTURE COURSE

Three years ago the traditional course of our predecessors consisted of a series of lectures supplemented by theoretical and practical programming exercises. The course itself introduces the basic concepts of computer architecture. The assembler programming language is a primary subject of the course and therefore programming exercises are required.

The Motorola 68000 processor family has a rather clean machine programming concept (e.g. linear addressing) compared to other CPUs. The practical exercises were therefore based on programming a M68k micro-computer module (see [7] for a similar module). Since the main purpose of these modules was the use in education, they had very limited video support and I/O facilities. Only ten stand-alone systems were available within a special room without network connection for data transfer to other computers. For saving their individual data, flash memory cards have been handed out to the students. When we were taking over the course, the number of students was growing significantly to nearly 100 due to the increasing number of computer science beginners. Since the students could only perform the exercises at such a system in the particular laboratory, about ten students had to share a single machine. Consequently during working hours the exercise room was always overcrowded and a lot of the students had to perform the exercises in the evening or early morning.

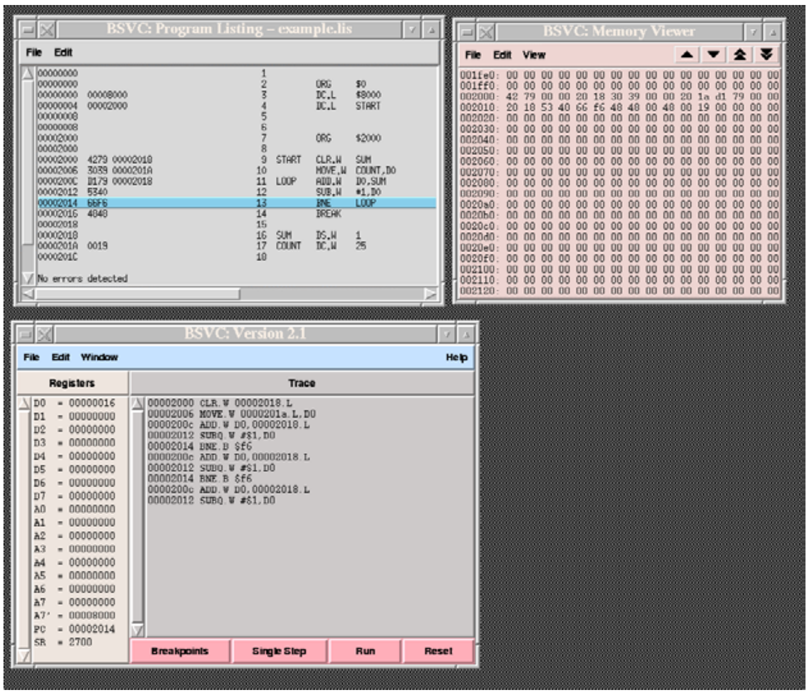
The students had to deliver a program listing to the tutors, who, based on the listing, decided whether the program fulfilled the required functionality or not. Of course, this was not an easy task. Another problem was the age of the used hardware equipment. Hardware failures (micro-computer and memory card) became more frequent, but no replacement could be purchased from the manufacturer. One solution to the problem would have been to buy a sufficient number of new micro-computer systems, but neither rooms nor fundings were available. In addition, we did not want to change the processor platform since the lectures and the lecture material is based on the Motorola 68k microprocessor family. Other directions resulting in the development of virtual exercises were required.

3. Virtual Computer Architecture Course

First, we enabled students to participate in lectures and exercise lessons virtually by providing the available teaching material (lecture slides and exercise sheets) on the web. In addition, students were able to submit the solutions via email to the tutors and access web pages containing information about their successful exercise participation. Although the first step was rather simple to achieve, it had several advantages: Students were able to easily keep track with lecture slide changes. Many students, in particular those with computer science as a minor subject, could not regularly attend the course due to lectures of their major subject that took place at the same time. Also, temporarily absent students, e.g. due to military or professional reasons, could further participate in the course. The major challenge, however, have been the laboratory exercises.

3.1 Simulation

A first observation was, that many of the practical exercises can be performed using a microprocessor simulation and assembler tool running on a Solaris workstation or on a Windows PC which is a significant relief to the students. Workstations or PCs provide a more convenient user interface than micro-computer systems. Using a high-end graphical text editor is much more comfortable than the simple line-based editor provided by micro-computer systems. Even more important for beginners is that runtime programming errors, which occur frequently in assembler programming, do not cause the whole computer to freeze and – as a result – loose all unsaved data, but the simulation framework provides better ways to recover from errors. Also several graphical debugging, tracing and monitoring tools are available to the students.



```
00000000                         1
00000000                         2              ORG     $0
00000000  00008000               3              DC.L    $8000
00000004  00002000               4              DC.L    START
00000008                         5
00000008                         6
00002000                         7              ORG     $2000
00002000                         8
00002000  4279 00002018          9     START    CLR.W   SUM
00002006  3039 0000201A         10              MOVE.W  COUNT,D0
0000200C  D179 00002018         11     LOOP     ADD.W   D0,SUM
00002012  5340                  12              SUB.W   #1,D0
00002014  66F6                  13              BNE     LOOP
00002016  4848                  14              BREAK
00002018                        15
00002018                        16     SUM      DS.W    1
0000201A  0019                  17     COUNT    DC.W    25
0000201C                        18
No errors detected
```

```
001fe0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
001ff0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
002000: 42 79 00 00 20 18 30 39 00 00 20 1a d1 79 00 00
002010: 20 18 53 40 66 f6 48 48 00 48 00 19 00 00 00 00
002020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
002030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
002040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
002050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
002060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
002070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
002080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
002090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0020a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0020b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0020c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0020d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0020e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0020f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
002100: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
002110: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
002120: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
D0  = 00000016
D1  = 00000000
D2  = 00000000
D3  = 00000000
D4  = 00000000
D5  = 00000000
D6  = 00000000
D7  = 00000000
A0  = 00000000
A1  = 00000000
A2  = 00000000
A3  = 00000000
A4  = 00000000
A5  = 00000000
A6  = 00000000
A7  = 00000000
A7' = 00008000
PC  = 00002014
SR  = 2700
```

```
00002000 CLR.W 00002018.L
00002006 MOVE.W 0000201a.L,D0
0000200c ADD.W D0,00002018.L
00002012 SUBQ.W #$1,D0
00002014 BNE.B $f6
0000200c ADD.W D0,00002018.L
00002012 SUBQ.W #$1,D0
00002014 BNE.B $f6
0000200c ADD.W D0,00002018.L
00002012 SUBQ.W #$1,D0
```

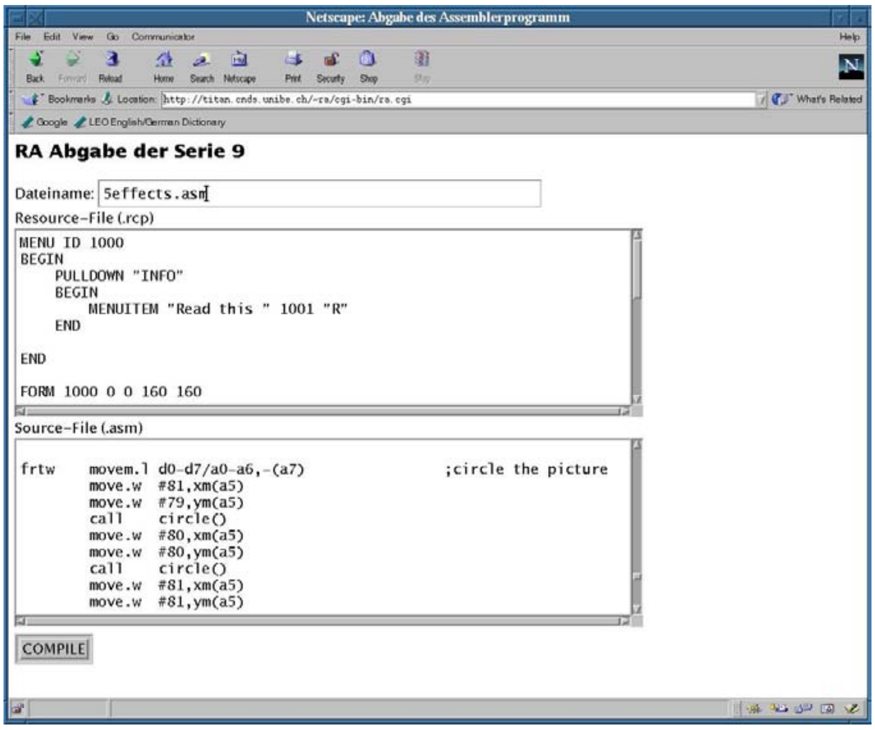
Figure 1 : M68k simulator and assembler tool

For our purpose, we selected the BSVC [1] microprocessor simulation framework developed at North Carolina State University. Originally written for UNIX, it has been ported to Windows. It includes a M68k simulator/assembler and provides a graphical user interface.

The availability of a simulator tool allows the students to perform many of the assembler programming exercises either on their private PC at home or at one of the numerous Solaris workstations at our institute (cf. Figure 1). The tutors are able to check the correctness of the exercises / programs by running the code on the simulator. This is a major improvement compared to the prior situation when the correctness of the source code had to be verified manually.

3.2 Cross Assembling

Nevertheless, the use of a simulation tool was not an optimal solution. Students should also have a real computer system that is able to run the developed assembler code. In addition, several exercises addressing the topic of input and output could not be performed with the simulator due to the lack of certain input/output functionalities. Therefore, we decided to identify a micro-computer system appropriate for running M68k code. Fortunately, some personal digital assistants (PDA) such as the Palm Top family have an M68k processor. Not wanting to miss the advantages of developing at a workstation we needed a cross-assembling software for the M68k. Such an assembler code development framework is called Pila [2].


Netscape: Abgabe des Assemblerprogramm
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://titan.cnds.unibe.ch/~ra/cgi-bin/ra.cgi What's Related
Google LEO English/German Dictionary
COMPILE

RA Abgabe der Serie 9

Dateiname: 5effects.asm

Resource-File (.rcp)

```
MENU ID 1000
BEGIN
    PULLDOWN "INFO"
    BEGIN
        MENUITEM "Read this " 1001 "R"
    END

END

FORM 1000 0 0 160 160
```

Source-File (.asm)

```
frtw    movem.l d0-d7/a0-a6,-(a7)               ;circle the picture
        move.w  #81,xm(a5)
        move.w  #79,ym(a5)
        call    circle()
        move.w  #80,xm(a5)
        move.w  #80,ym(a5)
        call    circle()
        move.w  #81,xm(a5)
        move.w  #81,ym(a5)
```

Figure 2 : Web Interface of the Cross-Assembler

As a speciality of the graphical PalmOS operating system the students have not only to provide assembler source code but also a resource file specifying the graphical resources of the program in a human-readable text format. In order to create the PalmOS binary file, the resource file has to be compiled first before assembling the source file. The cross-assembler combines the resources and the machine code into a single file - the Palm executable. One problem of Pila is, that it is available for both Windows and Linux but does not run on Solaris. Therefore, Pila allows to develop assembler programs on private student PCs running Linux or Windows but does not allow to do the same on the institute's student workstations. Since we have to assume that a student has only access to the Solaris student workstations another solution had to be found. That solution is based on a Linux server that acts as an assembling machine producing machine code for the PDA. A student has to provide an assembler file plus a resource file. The Linux server running the assembler environment then generates the executable file to be downloaded to the PDA. For that purpose, a cgi script has been developed that compiles the resource file, assembles the assembler file and generates the binary program file. The student accesses a web page [3] and has to enter the resource file and the assembler file (see Figure 2). By pressing a compile button, the web server runs the CGI script and generates the binary executable file. This file can then be downloaded to the local computer of the student. It can then be transferred to the PDA using available synchronisation programs (Figure 3). The web-based and therefore platform-independent solution avoids that students have to install the assembler programming environment on their own PCs.

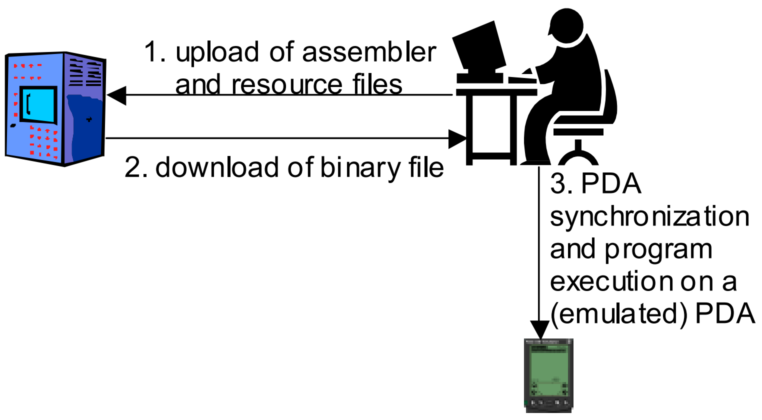

1. upload of assembler and resource files
2. download of binary file
3. PDA synchronization and program execution on a (emulated) PDA

Figure 3 : Assembler Code Development Steps

3.3 PDA Hardware and Emulator

In order to allow the students to run their developed programs we have bought a set of PDAs. Students can borrow these but also use an own PDA if available. However, the last step, i.e. running the program on a PDA can be performed completely virtual as well. For this purpose, one can use the Palm OS Emulator [4] that is available for many platforms such as Windows, MacOS, Linux, and Solaris. Alternatively, a student can use the available system on the student workstations. Once a correct program has been developed, students can email the source file to the tutors for evaluation / correction.

4. Conclusions and Outlook

This paper described the motivation and realisation of an virtual exercise for a computer architecture lecture. A student can perform assembler programming exercises without having a corresponding micro-computer system and without installing an assembler programming environment on her/his local computer. This enables students to attend to the complete course virtually although we keep the traditional lectures and theoretical exercise lessons. We consider our developments rather as a complementary service to increase the flexibility of the students.

5. References

[1] BSVC, www.redlinelabs.com/bsvc/
[2] Pilot Development Tools and Articles, www.massena.com/darrin/pilot/tanda.htm
[3] Übungen Rechnerarchitektur, titan.cnds.unibe.ch/~ra/cgi-bin/ra.cgi
[4] Palm OS Emulator (POSE): www.palmos.com/dev/tech/tools/emulator
[5] Nanoworld: www.nano-world.org
[6] VITELS: www.vitels.ch
[7] L.J.Electronics http://www.ljgroup.com/lj_tech/us/Sec_2/ET_D3/ljs_8_5.html