Application of WINDLX Simulator in Teaching Practice to Solve the Structural and Control Related in the Pipeline

Li Jingmei(✉), Wu Yanxia, Zhang Guoyin, Men Chaoguang, Ma Chunguang, Li Xiang, and Shi Changting

College of Computer Science and Technology, Harbin Engineering University, Nantong Avenue. 145, Harbin, China
lijingmei@hrbeu.edu.cn

Abstract. In the actual teaching of course Computer Architecture, such as parallel execution of instructions in the pipeline mechanism, as well as the three typical pipeline-related that affect the performance and the performance improvement after the elimination of three performance-related and other content, some students understand relatively difficult. WINDLX simulator not only can demonstrate the working principle of DLX instruction in the pipeline machine through the program written, but also can provide the statistic of program status which is executing on the pipeline machine.

Keywords: WINDLX simulator · Computer architecture · Related Instruction set

1 Introduction

As an important branch of computer science, computer architecture include the following aspects: Instruction system, data representation, addressing of operands, definition of registers, definition of terminal structure and exception conditions, storage system and management, I/O structure, definition and switching of machine working state, based on the above aspects and then study the internal computer software and hardware functions of the distribution mechanism, and ultimately to achieve the purpose of efficient and orderly system to complete the task. We hope we can point out and analyze the existing problems in existing disciplines, and plan the future development trend of disciplines. However, there are many problems in the actual teaching, for example: many concepts and principles of the content is not accompanied by the image of the intuitive experimental teaching environment, students can not clear learning objectives, the curriculum knowledge to practical application is difficult. Therefore, we adopt a combination of theoretical teaching and practical teaching. The introduction of DLX simulator to the actual teaching, achieve a win-win situation in both teaching and learning to.

S. Liu et al. (Eds.): eLEOT 2018, LNICST 243, pp. 47–52, 2018.
https://doi.org/10.1007/978-3-319-93719-9_7

2 DLX Instruction Set Structure

DLX is a pipeline example that runs through this topic, many discussions in the course, the simulation is based on DLX. DLX is a multivariate unsaturated instruction set structure, have a simple Load/Store instruction set, second, pay attention to instruction flow efficiency, third, simplifying instruction decoding, efficient support for the compiler.

DLX can deal with data types of 8-bit bytes, 16-bit halfwords, 32-bit integer numbers, and 32-bit single-precision floating-point numbers and 64-bit double-precision floating point numbers. The reason why there are 8 bytes in the recommended minimum data type is because the DLX needs to process character data. The reason why there is 16 halfwords, because it appears in a language similar to C, half-word data types are also very popular in the operating system code, after all, for these codes, the speed and length are equally important. The operation of the DLX is mainly for 32-bit integers and 32-bit or 64-bit floating-point numbers. When a byte or halfword is loaded into a 32-bit register, the upper bits of the 32-bit register are filled with zeros or sign bits and, once loaded into registers, are calculated as 32-bit integers.

3 WINDLX Simulator Introduction

The entire WINDLX folder contains WINDLX.EXE and WINDLX.HLP files, also need some assembly code files with. s extensions. When creating the installation directory for WINDLX, chinese characters can not appear in the path, to prevent the occurrence of some of its features can not be used. Double-click WINDLX.EXE start WINDLX platform. Under the interface of WINDLX, there are six sub-windows: Register, Code, Pipeline, Clock Cycle Diagram, Statistics and Breakpoints.

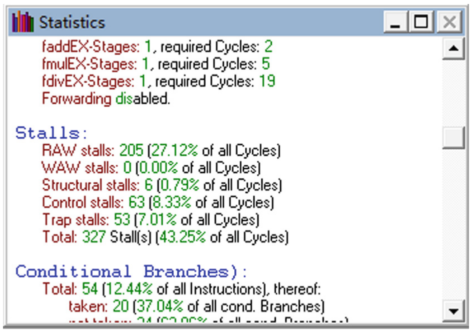
4 WINDLX Simulator Application

4.1 Structure Related and Processing Methods

Instructions in the process of overlapping implementation, the hardware resources can not meet the requirements of the implementation of overlapping instructions, resulting in hardware resource conflicts arising from the structural correlation. The second of these two instructions must wait until the previous one has been executed and can no longer be executed using the operator.

Computer hardware resources are limited, but also very expensive. In order to solve this phenomenon, there are many targeted simulators. Among the many simulators, WINDLX simulator has a very powerful hardware simulation capabilities. You can set the addition, multiplication and division of the units in the platform according to the needs of the experiment.

After the program is correctly run into the platform, you can find the number of structure related in the Statistics window (Statistics window), as shown in Fig. 1.



```
    faddEX-Stages: 1, required Cycles: 2
    fmulEX-Stages: 1, required Cycles: 5
    fdivEX-Stages: 1, required Cycles: 19
    Forwarding disabled.

Stalls:
    RAW stalls: 205 (27.12% of all Cycles)
    WAW stalls: 0 (0.00% of all Cycles)
    Structural stalls: 6 (0.79% of all Cycles)
    Control stalls: 63 (8.33% of all Cycles)
    Trap stalls: 53 (7.01% of all Cycles)
    Total: 327 Stall(s) (43.25% of all Cycles)

Conditional Branches):
    Total: 54 (12.44% of all Instructions), thereof:
        taken: 20 (37.04% of all cond. Branches)
```

Fig. 1. Related statistical results window

Then, based on the original hardware resources, we see that the number of structural relations in the statistics window is 6 (Structural stalls). Take a piece of code in the bubble sort as an example, as shown in Fig. 2.

```
addf        f2,f10,f1
addf        f10,f1,f11
addf        f11,f1,f2
```

Fig. 2. Generates structure-related code segments

The execution of this statement in Fig. 2. requires a continuous floating-point addition, Because there is only one adder, so the program execution to the three instructions generated structurally related, as shown in Fig. 3.

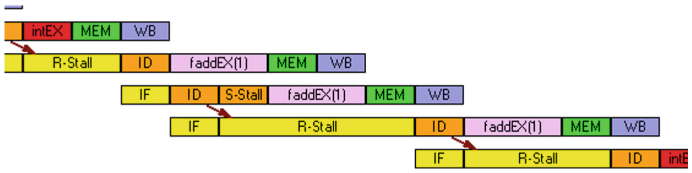
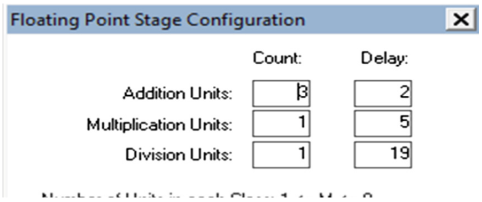


Fig. 3. Produces structure-related pipeline spatiotemporal plots

In the WINDLX platform, eliminating the number of structures related to the operation is very simple, reset the platform, reload the program to the platform, navigate to the Configuration option in the above navigation, click and select the Floating Point Stage Configuration function bar, the number of Addition Units (Addition Units) rewritten into 3, the effect shown in Fig. 4.

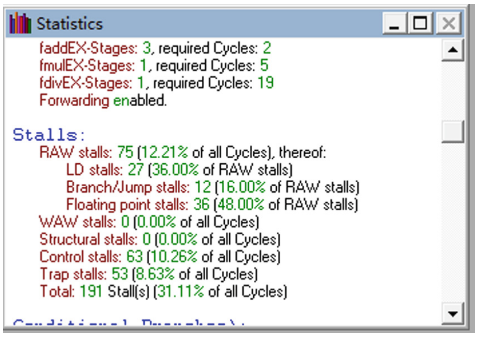


Floating Point Stage Configuration

	Count:	Delay:
Addition Units:	3	2
Multiplication Units:	1	5
Division Units:	1	19

Fig. 4. Add addition unit

After running the pre-loaded program again, this time enlarge the statistics window below the platform (as shown in Fig. 5.), After solving the structure of the code shown in Fig. 2 pipeline space-time map shown in Fig. 6.



```
Statistics
    faddEX-Stages: 3, required Cycles: 2
    fmulEX-Stages: 1, required Cycles: 5
    fdivEX-Stages: 1, required Cycles: 19
    Forwarding enabled.

Stalls:
    RAW stalls: 75 (12.21% of all Cycles), thereof:
        LD stalls: 27 (36.00% of RAW stalls)
        Branch/Jump stalls: 12 (16.00% of RAW stalls)
        Floating point stalls: 36 (48.00% of RAW stalls)
    WAW stalls: 0 (0.00% of all Cycles)
    Structural stalls: 0 (0.00% of all Cycles)
    Control stalls: 63 (10.26% of all Cycles)
    Trap stalls: 53 (8.63% of all Cycles)
    Total: 191 Stall(s) (31.11% of all Cycles)
```

Fig. 5. Statistics window

After running the pre-loaded program again, this time enlarge the statistics window below the platform (as shown in Fig. 5). After solving the structure of the code shown in Fig. 2 pipeline space-time map shown in Fig. 6.

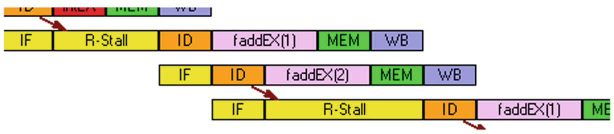


Fig. 6. Produces structure-related pipeline time-space map

By changing the number of floating-point adders, all three instructions in Fig. 2 can be performed on different adders, eliminating the structural dependencies of the code, according to the program's chart, it can be seen that the structural stalls have become zero. Through the WINDLX platform, we can virtually increase the number of addition units in the hardware, and use the statistics window to clearly see the sharp contrast before and after dealing with the structure, which is something that can not be done without the introduction of the platform.

4.2 Control Related and Processing Methods

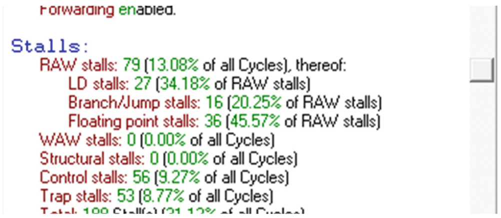
Control-related means that the instruction read in advance enters the value phase of failure due to various branch instructions (conditional branch instructions, rotor program instructions and interrupts, etc.) resulting in a jump in the program. Since the control-related effects are much larger in scope than the data and may cause changes in the direction of execution of the program, the control correlation may also be referred to as global correlation.

In the case of a bubble sort program, all the sorted integers are output by loop one by one. Since the loop is used, a branch instruction is used and the control is unavoidably generated. In the control-related code, by copying the code in the loop body multiple times, the execution of the jump instruction is reduced. As shown in Fig. 7, the code fragment often need to jump back to the beginning of the output, resulting in the previous stage of the pipeline fetch phase of the code read failure, and then generate the control in the pipeline.

```
output:     subi      r8,r8,1
            lf        f2,   PrintfValue(r3)
            cvti2d    f0,f2
            sd        Printf,f0
            addi      r14,r0,PrintfPar
            trap      5
            addi      r3,r3,4
            beqz      r8,over
            j         output
```

Fig. 7. Generates control related code screenshots

In order not to affect the pipeline performance, the specific method is to copy the loop code in the output function twice to achieve the purpose of reducing the execution of the jump instruction. Then we reload the program, after the implementation of the statistical window shown in Fig. 8.



```
Forwarding enabled.

Stalls:
   RAW stalls: 79 (13.08% of all Cycles), thereof:
      LD stalls: 27 (34.18% of RAW stalls)
      Branch/Jump stalls: 16 (20.25% of RAW stalls)
      Floating point stalls: 36 (45.57% of RAW stalls)
   WAW stalls: 0 (0.00% of all Cycles)
   Structural stalls: 0 (0.00% of all Cycles)
   Control stalls: 56 (9.27% of all Cycles)
   Trap stalls: 53 (8.77% of all Cycles)
```

Fig. 8. Statistics window using the loop unrolling technique

Comparing the data in Fig. 1, we can see that the reduction of control stalls from the original 63 to the current 56 did exactly what we wanted. And this platform gives us such a visual intuition, truly a simple, quick and accurate purpose. Observe the number of control related and the total number of program clocks before and after comparison in the Statistics window (Statistics window).

5 Conclusion

In a modern society characterized by rapid development of informatization, we are constantly accepting information from all sources. Educational innovation is exactly what we need to see as it relates to the key issues in the development of education. In the future, we envisage and are more convinced that we will further combine theoretical teaching with curriculum experiments to further enhance students' autonomous learning ability and ability to accept class knowledge, truly happy to carry out the teaching and learning of knowledge, and strive to push the computer architecture course to a new level.

References

1. Zhang C., Liu, Y., Zhang, C.: "Computer Architecture" web course. EB/OL (2005). http://sse.tongji.edu.cn/arch-course/index.htm
2. Qu, D., Xue, J., Fan, T.: Application of heuristic teaching in teaching of "computer system structure". J. Liaoning Univ. Nat. Sci. Ed. **37**(3), 218–220 (2010)
3. Zhang, C., Wang, Z., Zhang, C., et al.: Computer Architecture. Higher Education Press, Beijing (2003)
4. Xuan, L., Liansheng, H.: Computer architecture software simulation technology. J. Electron. Eng. Softw. Eng. **19**(24), 57 (2016)
5. Hesson, M.: Computer simulator: an educational tool for computer architecture. Am. J. Appl. Sci. **3**(11), 2114–2121 (2006)
6. Application of micro-class in the teaching of basic computer in university. J. Comput. Knowl. Technol. Acad. Exch. **12**(1)157–159 (2016)
7. Sibo, C.: Research on simulation technology of computer architecture software. J. Low Carbon World **3**, 181–182 (2016)
8. Huang, Q., Chang L., Zhang De-Hua, et al.: Host-based trusted security architecture based on trusted computing base. J. Inf. Netw. Secur. (7), 78–84 (2016)
9. Endong, W., Jicheng, C., Hongwei, W., et al.: Status, challenges and prospects of computer architecture simulation technology. J. Microsoft Microcomput. Syst. **37**(1), 178–185 (2016)
10. Shixin, Z., Lei, Z.: Analysis of computer architecture software simulation technology. J. Comput. CD Softw. Appl. **1**, 84–85 (2015)
11. Louboutin, M., Lange, M., Herrmann, F.J., et al.: Performance prediction of finite-difference solvers for different computer architectures. J. Comput. Geosci. **105**, 148–157 (2017)
12. Decyk, V.K., Singh, T.V.: Particle-in-Cell algorithms for emerging computer architectures. J. Comput. Phys. Commun. **185**(3), 708–719 (2014)