A STUDY ON BISC (BUS INSTRUCTION SET COMPUTER) ARCHITECTURE AND BISC-1

Student Number: 98-34567 Name: Makiko Ichirou SUZUKI Supervisor: Makiko TANAKA

1 Introduction

The architectures of microprocessors in ordinary use today are the CISC, RISC, and VLIW versions [1]. The working speeds of these architectures are being increased by multi-stage pipelines, superscalar, branching estimation, and instruction reordering. However, introducing these functions makes the processors increasingly complicated, so it is actually difficult to make additional changes in processor functions according to needs. In case of VLIW processors, although they are simpler than CISC and RISC processors, it is still difficult since instructions of VLIW processors depend on their functions. Since in the field of signal processing or image processing, functions of a processor have to be changed for the purpose, such as JPEG or MPEG–1,2,4,7 or filtering, an architecture by which functions of a processor can be easily changed is needed.

In order to cope with this situation, Yamashita proposed a new architecture called *bus instruction set computer* (BISC) [4]. In this architecture, instructions are essentially limited to data transfers between registers by the internal bus in the processor. Then, the structure of BISC processors is very simple, and units in the processor work highly independently. For example, instructions are fetched to buffers independently with decoding and execution. Instruction decoding is only the decoding of binary numbers. Execution is only data transfer between registers.

In this paper, we explain the BISC architecture by describing the detail of BISC-1. We show experiments by using Dhrystone 2.1 and optimize the number of buffers in BISC-1.

2 Overview of BISC-1

BISC-1 is a 32-bit integer processor, designed as a prototype BISC processor with a doubled bus for two data transfers per cycle.

2.1 Instruction format of BISC-1

Instructions of BISC-1 are restricted to data transfer between registers through the internal bus essentially. So, while traditional CISC or RISC architectures specify operations in the operation field of an instruction word, BISC needs no operation field because it has only one kind of instruction. It needs to specify only two registers of source and destination. Instruction format of BISC-1 is shown in Fig. 1.

Bit 31 is called Mark Bit (M-bit). M-bit indicates a block separation regarding control flow. When M-bit is ‘0’, BISC-1 continues instruction fetch. When M-bit is ‘1’, it is the last instruction word and the following instruction fetch depends on the state of internal registers. Bit 30, called Immediate-Data Bit (I-bit), indicates whether the next word in I-cache is an immediate data. When I-bit is ‘0’, the next word is an instruction word, otherwise an immediate data. The other bits specify the source/destination registers for data transfers. Two words specify five sets of source/destination register. Bit 29–24 indicates the register code number of destination (D0) and Bit 23–18 indicates one of source (S0). Bit 17–6 (D1, S1) is also written the source/receive register number of the next instruction. Further instruction is written on Bit 5–0 (D2) and Bit 29–24 (S2) of the next word (if I-bit is ‘1’, the word after next).

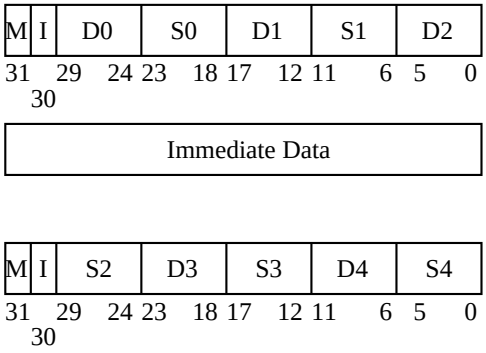

Figure 1: BISC-1 instruction format

2.2 Basic mechanism of BISC-1

The program is stored in I-cache. ICMCU fetches words from I-cache and transfers instructions to BCB or an immediate data to IDB. The address for I-cache access is specified by the program counter in ICMCU. Both BCB and IDB to receive instructions and data, respectively, are chosen by BCU.

One of BCB, which is chosen by BCU, receives 12-bit data to specify registers for data transfer from ICMCU. At the same time, one of BCB, which is chosen by BCU, sends 12-bit data as long as BCB has instructions.

In order to enhance the performance, BISC-1 has a doubled bus. Two data transfers can be performed with the doubled bus.

Each bus receives a data transfer instruction, which consists of a pair of 6-bit register code numbers, from BCB. Suppose they are *n* and *m*. Then, the data in the register *n* is transferred to the register *m* through a bus.

Table 1: Cycles and CPI for various number of buffers in BCB (with 2 buffers in IDB).

# of buffers	16	8	4	3
cycles	3345	3346	3347	3370
CPI	0.8119	0.8121	0.8124	0.8180

Table 2: Cycles, usage, and CPI for BISC-1 with 1 and 2 bus (with 4 buffers in BCB, 2 buffers in IDB).

multiplex degree of the bus		1	2
cycles		4638	3347
usage (%)	BUS1	88.83	79.53
	BUS2	–	43.56
CPI		1.1257	0.8124

The preceding instruction uses BUS1 and the following instruction uses BUS2. Only if the data in the register *m* can be read and the register *n* can be written on, then the data in the register *m* is transferred to the register *n*. After that, each bus receives the next instruction. Otherwise, the instructions are not executed in this cycle. A wired logic around a set of registers for a function unit decides locally whether read/write data is possible or not.

IDB is connected to IDR. When the data of IDR are transferred, the next data of IDB are sent to IDR at the same time, and one of IDB, which is chosen by BCU, can receive an immediate data from ICMCU.

3 Experimental Results

We examined the performance of BISC-1 with 16, 8, 4, and 3 buffers in BCB, and 8, 4, 3, and 2 buffers in IDB. In the first place, we investigated the relation between performance and the number of IDB. As the results of the examinations, it turned out that the number of cycles for Dhrystone 2.1 is constant with the number the buffers in IDB. Therefore, it can be concluded that two buffers in IDB is enough. In the second place, we varied the number of buffers in BCB with two buffers in IDB. The results are shown in Table 1. Cycles in Table 1 mean the number of clocks cycles needed for a main loop in Dhrystone 2.1. CPI is almost constant between 4 and 16 buffers, and CPI with 3 buffers is worse. Therefore, the optimal number of buffers in BCB is four. As a result, the optimal number of buffers in BCB is four and IDB is two, when using this memory system.

The multiplex degree of the bus means the number of executable instructions in a cycle. Therefore, the performance of BISC-1 can be improved by increasing the multiplex degree of the internal bus. So, we compare the doubled bus version with the single bus version. The result is given in Table2. CPI for the doubled bus is smaller than the single bus, so the double bus has an advantage.

Although the usage of BUS1 can be raised by optimization of the instruction stream, the usage of BUS2 is not high. So, it is necessary to use BUS2 more effectively.

It is not necessary to implement so many buffers in BCB and IDB. The reason would be that BCB and IDB have only to hold instructions and an immediate data for the next cycle when using this memory system.

From observation of the process, it was proved that unconditional jump can be performed with little delay. The reason is that the specifier of unconditional jump and the target address is set when data are transferred to OCR and BAR, and instructions after the branch can be fetched previously.

ALU is almost busy. In order to solve this problem, the operation unit has to be added.

4 Conclusions

In this paper we discussed BISC architecture and the structure, specifications, and features of BISC-1 designed as a prototype BISC processor. We also showed the experimental results of BISC-1 and optimized the multiplex degree of the internal bus and the number of buffers in BCB and IDB.

Performance could be improved, by increasing the number of buses and implementing a floating-point unit, a multiplication unit, and a division unit, according to the needs of an application domain. And in order to exploit the performance of BISC-1, we are developing the high performance compiler for BISC-1. We are also planning to implement efficient 'interrupt' of BISC processors.

References

[1] John L. Hennessy and David A. Patterson, "Computer Organization & Design:The Hardware/Software Interface," Morgan Kaufmann Publishers, San Francisco, 1994.

[2] G. Jack Lipovski, "The architecture of a simple, effective control processor," In Second Euromicro Symposium on Microprocessing and Microprogramming, pp.7-18, Oct. 1976.

[3] Henk Corporaal and Paul van der Arend, "Move32int, a sea of gates realization of a high performance transport triggered architecture," Microprocessing and Microprogramming, vol. 38, pp.53-60, Sept. 1993.

[4] Yukihiko Yamashita, "A New Architecture of Multi-Purpose Microprocessor –Bus Instruction Set Computer (BISC)–," Technical report of IEICE, no. CPSY 96-70, pp.9-14, Oct. 1996. (in Japanese)