



Course Code : SOF108

Course Name : Computer Architecture

Lecturer : Md Iftekhar Salam

Academic Session : 2021/04

Assessment Title : A study on the MIPS Instruction Set Architecture

Submission Due Date : 9<sup>th</sup> July 2021

Prepared by :	Student ID	Student Name
	SWE2009510	Liu Aofan
	SWE2009513	Su Yanyu
	SWE2009689	Ang Chen Jia
	SWE2009507	Lai Ren Jie
	SWE2009500	Hu Xinao

Date submitted : 9<sup>th</sup> July 2021

Component	Marks	Remarks
A study on the MIPS Instruction Set Architecture		
Total	/15	
%	/15	

### Own Work Declaration

I/We hereby understand my/our work would be checked for plagiarism or other misconduct, and the softcopy would be saved for future comparison(s).

I/We hereby confirm that all the references or sources of citations have been correctly listed or presented and I/we clearly understand the serious consequence caused by any intentional or unintentional misconduct.

This work is not made on any work of other students (past or present), and it has not been submitted to any other courses or institutions before.

Signature:

Liu Aofan

Lai Renjie

Hu Xinao

Yanyu Su

Ang B

Date: 1<sup>st</sup> July 2021

## Table of Contents

Abstract .....	5
Acknowledgment .....	6
Introduction.....	7
Research Methodology and Approach .....	8
Chapter 1 Theoretical Overview .....	9
1.1 The Basic Instruction Cycle.....	9
1.2 Different kinds of computer languages .....	10
1.2.1 High-level language .....	10
1.2.2 Machine Language.....	10
1.2.3 Assembly Language .....	11
1.2.4 Comparison between different languages .....	11
1.3 Instruction set architecture .....	12
Chapter 2 MIPS Instruction Set Architecture .....	14
2.1 Data operations, data transfer operations, sequencing.....	14
2.1.1 Data operations & data transfer: .....	14
2.1.2 Sequencing:.....	16
2.2 Addressing modes in MIPS.....	18
2.3 Instruction format, encoding, etc. ....	19
Chapter 3 Pipelining in MIPS .....	21
3.1 What is pipelining? .....	21
3.2 What is RISC (Reduced Instruction Set Computer)? .....	22
3.3 Stages of RISC pipelining.....	23
3.4 Pipelining Hazard.....	24
3.4.1 Resources Hazards .....	24
3.4.2 Data Hazards .....	25
3.4.3 Control Hazards .....	26

Chapter 4 MIPS Development .....	28
4.1 Application of MIPS .....	28
4.2 Memory Organization in MIPS .....	29
4.2.1 Memory Organization in Computer Architecture .....	29
4.2.2 Memory in MIPS .....	29
4.2.3 View memory as bytes or words .....	30
4.3 Comparison of MIPS with other ISAs .....	31
Chapter 5 Conclusion.....	33
Reference .....	34
Appendix 1 Contributions.....	37
Appendix 2 Record of meeting minutes .....	39
Appendix 3 Marking Rubrics.....	46
Appendix 4 Digital Receipt .....	47
Appendix 5 Similarity Report Summary .....	47

## **Abstract**

As one of the most classic streamlined instruction set architectures in the industry, MIPS was once considered to be comparable to Arm and x86, becoming one of the world's three major mainstream architectures. For a long time in the past, people thought that MIPS can keep pace with Arm and X86.

But now 40 years old MIPS is increasingly marginalized. After the open-source project was once again stranded, the future of this RISC pioneer has become the focus of many practitioners. Some people sigh: This is really another BlackBerry that has encountered Android.

However, as a classic architecture that has been verified for many years and has been produced in large quantities, there are a lot of materials and reference books to learn. It is still deserving respect. In this paper, we will give a brief introduction to MIPS.

## **Acknowledgment**

This project would not be possible without the help of many people, we would like to express our gratitude to my professors at the School at Xiamen University Malaysia, our lecturer Md Iftekhar Salam.

Here, we would like to extend our sincerest gratitude and heartfelt thanks to Md Iftekhar Salam. Thank you for your help and guidance on the writing of this thesis, and your own suggestions or comments on the improvement of the thesis. Especially your support in our spirit and study which we do value.

## **Introduction**

In 1984, John LeRoy Hennessy, former president of Stanford University, and his team founded MIPS. Their business model is to license the completed chip design to other manufacturers so that they can easily manufacture high-performance CPUs which makes MIPS become more and more popular during these years.

Looking back at history, MIPS was once brilliant in the 1990s. Nearly half of the large semiconductor companies adopted MIPS designs to manufacture chips. Of the over 16 billion microprocessors produced each year at that time, RISC processors dominant the market.

In addition, as a classic RISC architecture, MIPS has a lot of learning resources, which is very suitable for research so we want to learn more about MIPS and RISC.

In this article, we carry out a case study on the MIPS Instruction Set Architecture. First, we give a brief introduction to the Computer Organization and Architecture in the world today, including basic instruction cycle, different programming languages at different times, and Instruction Set Architecture.

Here is our research content:

- Research on the fundamental concepts of computer architecture and trying to analyze computer architecture and its system.
- Try to do research on parallel computer architecture and its structure.
- Knowledge about memory organization and pipelining in MIPS.
- Learning about the data operation in MIPS and how data was transferred.

## **Research Methodology and Approach**

In this article, the main research method we use is the literature survey, which uses extensive research literature to acquire relevant knowledge. Because of academic characteristics such as the rigor of papers and conference records, it is very likely that this method can help us effectively understand the research topic.

- Web search

Make full use of Web of Science, use SCI, EI, and use the keyword MIPS to find relevant documents in the past five years. First, we find the most relevant titles in the title and understand what the article explains by reading the abstract. If the paper is really useful, then start reading the main text.

- Follow up reading

During the research, if we find a paper that is very useful, we will try to find the latest relevant paper or review, and trace the most important paper forward from its references.

- Research authoritative data

Obtain data related to the topic we want to know by studying the announcements issued by scientific institutions and academic conferences

And it has the following benefits:

- Able to understand the historical background of the problem and help study the topic.
- It covers the limitations of time and space. Therefore, we are able to study a wide range of papers, which helps to understand the whole picture of things.



## Chapter 1 Theoretical Overview

### 1.1 The Basic Instruction Cycle

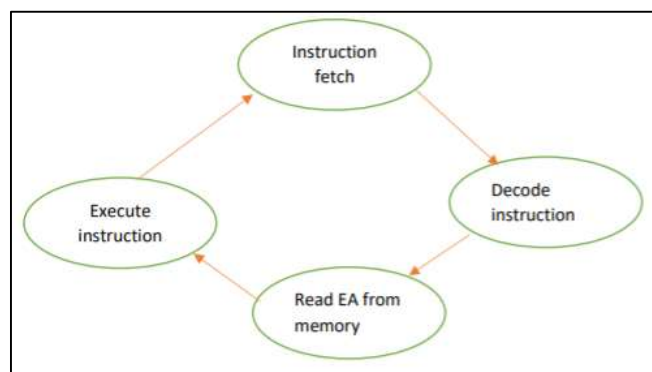
We usually think that the sum of the time from fetching an instruction from memory to executing it is an instruction cycle. So, we can also call it the fetch-decode-execute cycle. It means the CPU will perform fetch, decode and execute instruction sequentially.

The basic instruction cycle consists of the following stages which are illustrated in Table 1.1 - 1:

*Table 1.1 - 1 stages of instruction cycle*

Stage	Description
<b>Fetch</b>	In this stage, the processor fetches the instruction from memory. Usually has the following steps: Firstly, the address in the PC is transmitted to the MAR. Secondly, take the instruction from this address and save them in the MDR. Then, pass this instruction to IR. Next, after sending the operation code in IR to the CU, the address code in IR is transferred to MAR. Finally, fetch the data from the storage to the MDR, and send MDR data to ACC. [1].
<b>Decode</b>	In this stage, the processor will know what instruction is to be executed. The opcode will be decoded at this stage. If needed, the operands are retrieved from the addresses [2].
<b>Execute</b>	The control unit transfers the control signal to ALU. And the ALU executes the instruction.

In addition to the three stages mentioned above, reading effective address from memory is also a step. From Figure 1.1 - 1, we can see the simplified process of an instruction cycle.



*Figure 1.1 - 1 simplified process of an instruction cycle*

In a word, different machines have different requirements for the machine cycle. When we are programming, we should consider using fewer machine cycles to gain a faster execution rate.

## **1.2 Different kinds of computer languages**

### ***1.2.1 High-level language***

High-level language can be considered as a programming language. The programming languages we are familiar with such as C++, c, python are high-level programming languages. They have different syntaxes. Besides, their command formats are not the same, too.

High-level language's compiler compiles the source program into object code. The main difference between variables and registers in high-level languages is that the number of registers is limited. In the MIPS architecture, registers are 32 bits [1].

It makes programmers write instructions in languages that are easier to understand than a low-level language. However, before executing the code, it must be processed by the compiler. [2].

Convert the executable machine code of the computer into a high-level language program is very necessary. Because it can greatly help us analyze, understand, test, verify, transplant, and modify programs. This conversion process is called anti-editing.

### ***1.2.2 Machine Language***

The collection of machine instructions is called machine language. Unlike a high-level language, it can be directly recognized and executed by a computer without being compiled by a compiler. In addition, it has the characteristics of flexibility and fast speed.

Machine language does not include words or letters. The machine code composed of binary code can also be expressed in hexadecimal [3]. In terms of usage, it can be regarded as the lowest level language.

### ***1.2.3 Assembly Language***

Same as machine language, assembly language is also a low-level language. It can be applied in not only electronic computers, but also microcontrollers and microprocessors. It has the advantages of fast running speed and short target code generation. By learning assembly language, we can have a general understanding of the underlying operation of the machine [1].

Each instruction of MIPS assembly language has a fixed type and number of operands. Some instructions contain only a simple variable, while some instructions contain complex data structures like arrays. A data structure like an array will be stored in the memory. The operands of arithmetic instructions in MIPS instructions are limited to registers. Therefore, to complete the transfer of data between memory and registers, there must be a corresponding instruction [1].

Now, assembly language is not widely used for programming. It is required by embedded operating system drivers and real-time running programs. Besides, we can also see that it is often employed in hardware operations and some program majorization situations.

### ***1.2.4 Comparison between different languages***

All in all, we can know that high-level language is much easier to write and understand for programmers than two other languages. However, Machine language can be run directly by a computer, which is more efficient than another two. These three languages do not exist in isolation. When the assembler program runs, it can be turned into machine code first, and then it can run. Similarly, computers can only understand translated programs written in high-level languages. We can feel the difference better from the table below.

Table 1.2 - 1 High-level Language, Machine Language, and Assembly Language

	High-level Language	Machine Language	Assembly Language
<b>Introduction</b>	Including C ++, Java, Python, and other programming languages.	Represented by binary code.	Use specific names and symbols to represent operation codes. Both the names and symbols are not difficult to remember.
<b>Compile Requirement</b>	Yes	No	Yes
<b>Advantages</b>	Show data operation and program control structure readily. Write, debug and maintain easily. Describe various algorithms well. [4]	The computer can run it directly. Concise. The operation speed is fast.	It has high memory efficiency. Mainly hardware-oriented [5].
<b>Disadvantages</b>	The compiled code is long and the execution speed is slow. Cannot exchange directly with the hardware. [4]	Poor intuition. It is difficult to check and debug the program.	Difficult to understand. The syntax is not easy to remember.

### 1.3 Instruction set architecture

There are many layers between the application layer and the physics layer. Instruction set architecture (ISA) is one of them.

The instruction set architecture refers to a series of instructions needed to control a computer system, ranging from instruction format to addressing mode. Therefore, the ISA not only determines the capabilities required by the CPU but also determines the format of the instructions and the structure of the CPU.

X86 architecture and ARMv8 architecture are in the category of instruction set architecture. Usually, we talk about computer CPUs, which nearly are all based on X86 and X64 architectures; mobile phone CPUs are now all ARM architectures.

Moreover, instructions in the instruction set will include all the necessary instructions related to the data, these instructions have a close relationship with assembly language. In the meantime, these instructions can play the role of addressing, memory operation, and flow operations. Moreover, arithmetic and logic operations are

also included.

The instruction set which clearly states what the hardware needs to do is a collection of opcodes, such as addition, subtraction, and jump [7].

In general, the ISA is mainly divided into two categories, one is reduced instruction set computer (RISC), and the other is complex instruction set computer (CISC).

RISC mainly includes those instructions that are often used and provides some necessary instructions. And in contrast, CISC has complex instructions and uses a large number of flexible addressing methods to improve the running speed of the program. In the meantime, this can increase the speed of compilation.

Here are the differences between CISC and RISC (Table 1.3 - 1):

*Table 1.3 – 1 RISC vs. CISC*

	<b>CISC</b>	<b>RISC</b>
<b>Emphasis</b>	Hardware	Software
<b>Clock cycle</b>	Include multi clock cycle	Single clock cycle only
<b>Instruction design</b>	Include complex instructions	Relatively simple
<b>Addressing methods</b>	Supports multiple addressing methods	Support a few addressing methods
<b>Length of instruction sets</b>	Deals with unequal length instruction sets	Deals with equal length instruction sets

Common CISC architectures which are famous for their RISC architectures:

- X86. Owing to the commercial marketing of Intel processors, it has been widely used worldwide.
- ARM. Because some mobile phones of Apple, Nokia, and Samsung use ARM architecture, the emergence of ARM architecture has become common. The business model that relies on IP authorization has also won valuable opportunities for it [9].

- MIPS. It is more academic, but due to the fragmentation of the ecosystem, it is rare in household electronic devices and has a small market share [10].
- IBM power. This is a type of IBM RISC processor, which was mainly used in mid-to-high-end home computers and commercial servers at the time. And some Apple computers also use this architecture.

## **Chapter 2 MIPS Instruction Set Architecture**

### **2.1 Data operations, data transfer operations, sequencing**

#### ***2.1.1 Data operations & data transfer:***

Data operation is a collection used to describe the dynamic characteristics of the system and perform operations on various objects or instances in the database. It mainly includes two kinds of operations: query and modification and defines the exact meaning of these operations and the implementation method of these operations through the data model.

In MIPS instruction set architecture, a data operation mainly refers to a series of processing (operations) of data and instructions in the system composed of the instruction cache, program counter, register file, memory, control logic, and arithmetic logic unit (ALU).

In a data operation performed in MIPS, to start with, the program counter saves the address of acquired instructions, and then the instructions are extracted from the memory to the register. Next, the instructions are decoded by the control logic, and it informs the arithmetic logic unit and the register file of the operation to be performed. The arithmetic logic unit then executes the instructions, and the execution results return back to the register. Last, the control logic renews the counter to process the upcoming instruction. At this point, the data operation flow to an end.

Broadly speaking, data transfer refers to the transmission of data information from one location to another via a certain communication method, such as sending and receiving data signals using binary codes. In MIPS, the flow of data transfer is as follows: At the beginning, the arithmetic logic unit generates an instruction address and the address enters the address register. Then the results of entering & leaving the memory are saved in the data register. Finally, all relevant data will be put into the register file.

Next, we will show a more detailed MIPS data operation and transmission

process. As shown in Figure 2.1 - 1 & Table 2.1 - 2.

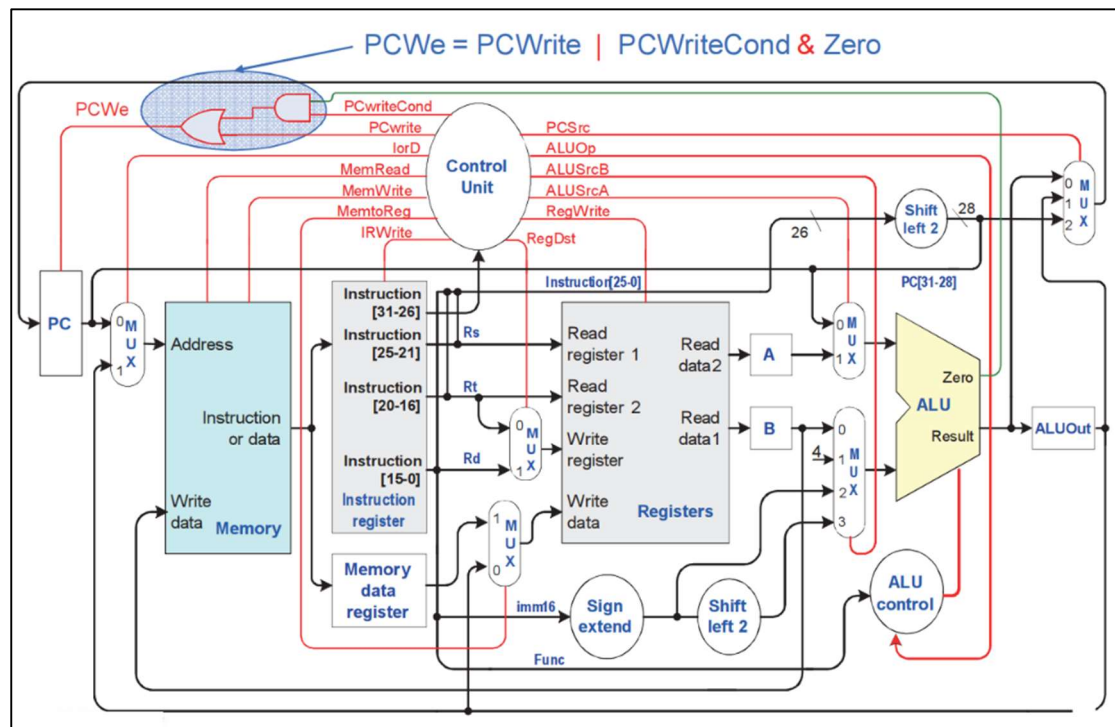


Figure 2.1 - 1 MIPS Instruction Set Architecture

Table 2.1 - 1 Control Signals and Functions

Control Signal	Value (Binary)	Function
ALUOp	00	The ALU performs addition
	01	The ALU performs addition
	10	The ALU operation is determined by the Func field (IR [5:0])
ALUSrcB	00	ALU input port B data source, select register B
	01	ALU input port B data source, select constant 4
	10	The ALU input port B selects the sign extension output
	11	The ALU input port B selects the shifter output
PCSrc	00	PC input source selects ALU output
	01	PC input source selects register ALU Out
	10	PC input source selects transfer address

#### Detailed analysis of the execution process:

① The main task of this stage is to fetch instructions from memory and calculate the address of the next instruction. At first, the controller sets the control signals IRWrite & MemRead to be valid and sets IorD to 0 to select PC as the memory address source. Then, the controller sets the control signals ALUSrcA as 0, ALUSrcB as 01, and ALUOp as 00 (addition operation), so that ALU can realize the calculation

of PC+4 and sets PCSrc as 00. The address of the next instruction will be successfully stored.

② The main task of this stage is to translate instructions and prepare data. It is worth noting that neither stage 1 nor stage 2 can know the specific content of instruction. First, read registers Rs & Rt and store them in registers A and B respectively (if it is a branch instruction, the address of PC may be used in the next cycle). Then the offset address is prepared, and the controller sets ALUSrcB as 11, ALUSrcA as 0, and ALUOp as 00. In this way, the offset address is calculated and stored in the register ALUOut.

③ The main task of this stage is to complete specific instruction operations or prepare instruction data. The first step is to access the memory instruction. The controller sets ALUSrcA as 1, ALUSrcB as 10, and ALUOp as 00. The ALU adds the operands to obtain the memory address and stores it in ALUOut. The second step is the operation instruction. The controller sets ALUSrc as 1, ALUSrcB as 00, and ALUOp as 10, completes the arithmetic operation and saves the temporary result in ALUOut. The third step is called the branch instruction. The controller sets ALUSrcA as 1, ALUSrcB as 00, and ALUOp as 01 for the equivalence test. The fourth step is the jump instruction. The controller sets pcsrc as 10, pcwrite as 1, and stores the jump address into PC after passing through an OR gate.

④ At first, the instruction is operated. The controller sets RegDst as 1, MemtoReg as 0, and RegWrite as 1, and stores the calculation results in the register Rd. Then, the instruction is accessed and stored. If it is a word storage instruction, the data is directly written into the memory. If it is a word fetching instruction, read out the data that come from the memory and put it into MDR. The controller sets IorD as 1 and MemRead as 1, to directly store data into memory or fetch data from memory to MDR.

### ***2.1.2 Sequencing:***

In MIPS instruction set architecture, the sequencing instruction is responsible for making decisions (such as whether to execute the next instruction, what instruction to execute next, etc.). Figuratively speaking, sequencing instructions are just "forks", which change the control flow of the whole program. The followings are some examples of MIPS sequencing instructions:

#### **① For loop: C code**



```
int main()
{
    int sum = 0;
    for(int i=0; i<=n; ++i)
    {
        sum = sum + i;
    }
    return 0;
}
```

Figure 2.1 - 2 For loop in C code

**For loop: MIPS code**

```
# $s0 == Sum, $s1 == n, $t0 == i
move $s0, $zero # register assignment
lw $s1, n # assume global symbol
li $t0, 1 # literal assignment
loop: beq $t0, $s1, done # loop test
      add $s0, $s0, $t0 # Sum = Sum + i
      addi $t0, $t0, 1 # ++i
      b loop # restart loop
done:
```

Figure 2.1 - 3 For loop in MIPS

② **Conditional control structure: C code**

```
if(a < b)
    goto A;
else
    goto B;
```

Figure 2.1 – 4 Conditional control in C code

**Conditional control structure: MIPS code**

```
# $s3 == a, $s4 == b
slt $t1, $s3, $s4
beq $zero, $t1, B
A:  # code...
    b C
B:  # code...
C:
```

Figure 2.1 - 5 Conditional control in MIPS

③ **Conditional branch instructions: C code**

```
if(i == j) {
    h == i + j;
}
```

Figure 2.1 – 6 Conditional branch in C code

**Conditional branch instructions: MIPS code**

```

bne $t0, $t1, <label> # branch on not-equal
                        # PC += 4 + Label if
                        # $t0 != $t1
beq $t0, $t1, <label> # branch on equal

# starts from 1000
Loop: add $t1, $s3, $s3
      add $t1, $t1, $t1
      add $t1, $t1, $s6
      lw  $t0, 0($t1)
      bne $t0, $s5, Exit
      add $s3, $s3, $s4
      j  Loop
Exit:

```

Figure 2.1 - 7 Conditional branch in MIPS

④ Unconditional branch instructions: MIPS code

j	Label	# PC = Label
b	Label	# PC = Label
jr	\$ra	# PC = \$ra

Figure 2.1 - 8 Unconditional branch in MIPS

## 2.2 Addressing modes in MIPS

Addressing modes usually refer to the ways of specifying a memory address or an operand. The hardware of MIPS supports only one memory addressing mode: register base address + immediate offset (16-bit address offset). Particularly, the address offset must be between -32768 and 32767 (16 bits, that is, 2 to the 16th power). Any load and store instruction can be written as LW \$1, offset (\$2), and the register can be used as source operand or destination operand. It is important to note that data access in memory must be strictly aligned (at least 4 bytes aligned). The MIPS assembler can also use synthetic instructions to support multiple addressing modes, as follows:

- ① Constant addressing: using 32-bit constant addressing.
- ② Register indirect addressing: register address + offset (offset is 0).
- ③ Direct addressing(almost): addressing by external variable name or data label.
- ④ Direct + index: the label address specified by the register + offset.

⑤ Global pointer (Gp) relative addressing (data access).

⑥ PC-relative addressing: Based on the current instruction and offset value from immediate value we now the next instruction.

⑦ Immediate addressing: operands are embedded in coded instructions.

\*It is worth noting that addressing mode is not an instruction type, and there are essential differences between them: addressing mode is the way to determine the address (register or memory), while the instruction type is how to put instructions together.

### 2.3 Instruction format, encoding, etc.

There are only three instruction formats of MIPS, which are J-type, R-type, and I-type:

①J(jump) type: also known as "hard jump instruction", this type of instruction uses a 26-bit immediate number as the jump target address, plus 2-bit alignment bits, which can address a 28-bit space (256M). The main function of this instruction format is to return from exceptions. Table 2.3 -1 illustrates the J-type instructions in MIPS.

***Op(6-bit) + target-address(26-bit)***

*Table 2.3 - 1 J-type Instruction*

Symbol	Instruction Format		Example	Addressing Mode
<b>J-type</b>	op(6)	address(16)		
j	00 00 10	address	j 10 00 0	Pseudo direct addressing
jal	00 00 11	address	jal 10 00 0	Pseudo direct addressing

②R(register) type: this type of instruction reads two source operands and then writes the calculation result back to the register file. The main function of this instruction format is to perform a register-register ALU operation [6]. In Table 2.3 – 2 we try to show the R-type instructions in MIPS

***Op(6-bit) + Rs (The first source operand register: 5-bit) + Rt (The second source operand register: 5-bit) + Rd (The destination operation register for storing results: 5bit) + Shamt (Offset, used to shift instructions: 5-bit) + Funct (Function to supplement the op-code: 6-bit)***

*Table 2.3 - 2 R-type Instruction*

Symbol	Instruction Format						Example	Addressing Mode
<b>R-type</b>	op(6)	rs(5)	rt(5)	rd(5)	shamt(5)	funct(6)		
add	00 00 00	rs	rt	rd	00 00 0	10 00 00	add \$1,\$2,\$3	register addressing
sub	00 00 00	rs	rt	rd	00 00 0	10 00 10	sub \$1,\$2,\$3	register addressing
and	00 00 00	rs	rt	rd	00 00 0	10 01 00	and \$1,\$2,\$3	register addressing
or	00 00 00	rs	rt	rd	00 00 0	10 01 01	or \$1,\$2,\$3	register addressing
nor	00 00 00	rs	rt	rd	00 00 0	10 01 11	nor \$1,\$2,\$3	register addressing
slt	00 00 00	rs	rt	rd	00 00 0	10 10 10	slt \$1,\$2,\$3	register addressing
sltu	00 00 00	rs	rt	rd	00 00 0	10 10 11	sltu \$1,\$2,\$3	register addressing
sll	00 00 00	00 00 0	rt	rd	shamt	00 00 00	sll \$1,\$2,10	register addressing
srl	00 00 00	00 00 0	rt	rd	shamt	00 00 10	srl \$1,\$2,10	register addressing
jr	00 00 00	rs	0 00 00	0 00 00	00 00 0	00 10 00	jr \$3 1	register addressing

③I(immediate) type: this type of instruction uses a 16-bit immediate number as the source operand. The main function of this instruction format is to load/store bytes. Conditional branching, jumping, and linking registers and special registers for reading and writing. Below are the I-type instructions in MIPS.

$$Op(6\text{-bit}) + Rs(5\text{-bit}) + Rt(5\text{-bit}) + Address/Immediate(16\text{-bit})$$

*Table 2.3 - 3 I-type Instruction*

Symbol	Instruction Format				Example	Addressing Mode
<b>I-type</b>	op(6)	rs(5)	rt(5)	immediate(16)		
addi	00 10 00	rs	rt	immediate	add \$1,\$2,100	Immediate addressing
lw	10 00 11	rs	rt	address	lw \$1,10(\$2)	Base address addressing
sw	10 10 11	rs	rt	address	sw \$1,10(\$2)	Base address addressing
lh		rs	rt	address	lh \$1,10(\$2)	Base address addressing
sh		rs	rt	address	sh \$1,10(\$2)	Base address addressing
lb		rs	rt	address	lb \$1,10(\$2)	Base address addressing
lbu		rs	rt	address	lbu \$1,10(\$2)	Base address addressing
sb		rs	rt	address	sb \$1,10(\$2)	Base address addressing
ll		rs	rt	address	ll \$1,10(\$2)	Base address addressing
sc		rs	rt	address	sc \$1,10(\$2)	Base address addressing
lui		rs	rt	immediate	lui \$1,20	Immediate addressing
andi		rs	rt	immediate	andi \$1,\$2,20	Immediate addressing
ori	10 10 11	rs	rt	immediate	ori \$1,\$2,20	Immediate addressing

The following table tells us that differences and common in types of instructions

*Table 2.3 - 4 Types of Instructions (MIPS)*

Function	Instruction	Effect
add	add R1,R2,R3	R1=R2 + R3
sub	sub R1,R2,R3	R1=R2 - R3
add immediate	addi R1,R2,145	R1=R2 + 145
multiply	mult R2,R3	hi, lo=R1 * R2
divide	div R2,R3	low=R2/R3, hi=remainder
and	and R1,R2,R3	R1=R2 & R3
or	or R1,R2,R3	R1=R2   R3
and immediate	andi R1,R2,145	R1=R2 & 145
or immediate	ori R1,R2,145	R1=R2   145
shift left logical	sll R1,R2,7	R1=R2 << 7
shift right logical	srl R1,R2,7	R1=R2 >> 7
load word	lw R1, 145(R2)	R1=memory[R2 + 145]
store word	sw R1, 145(R2)	memory[R2 + 145]=R1
load upper immediate	lui R1, 145	R1=145 << 16
branch on equal	beq R1,R2,145	if(R1==R2) go to PC + 4 + 145*4
branch on not equal	bne R1,R2,145	if(R1!=R2) go to PC + 4 + 145*4
set on less than	slt R1,R2,R3	if(R2<R3) R1=1, else R1=0
set less than immediate	slti R1,R2,145	if(R2<145) R1=1, else R1=0
jump	j 145	go to 145
jump register	jr R31	go to R31
jump and link	jal 145	R31=PC + 4; go to 145

Finally, we make a table to differentiate these three kinds of instruction formats.

*Table 2.3 - 5 Three Instruction Formats of MIPS*

Name	Fields						Comments
Field size	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	All MIPS instructions are 32 bits long
R-format	op	rs	rt	rd	shamt	funct	Arithmetic instruction format
I-format	op	rs	rt	address/immediate			Transfer, branch, imm. format
J-format	op	target address					Jump instruction format

Use MIPS instruction format for arithmetic operation, for example: subtract and add both have three operands (Including two source operands and a result) [7].

**sub a, b, c # a gets b - c.**

**add a, b, c # a gets b + c.**

All common arithmetic operations using MIPS instruction format have this form. In figure 9, some instructions (including J-type, R-type, and I-type instructions) and their meanings(effect) are also illustrated in detail in the form of tables.

## Chapter 3 Pipelining in MIPS

### 3.1 What is pipelining?

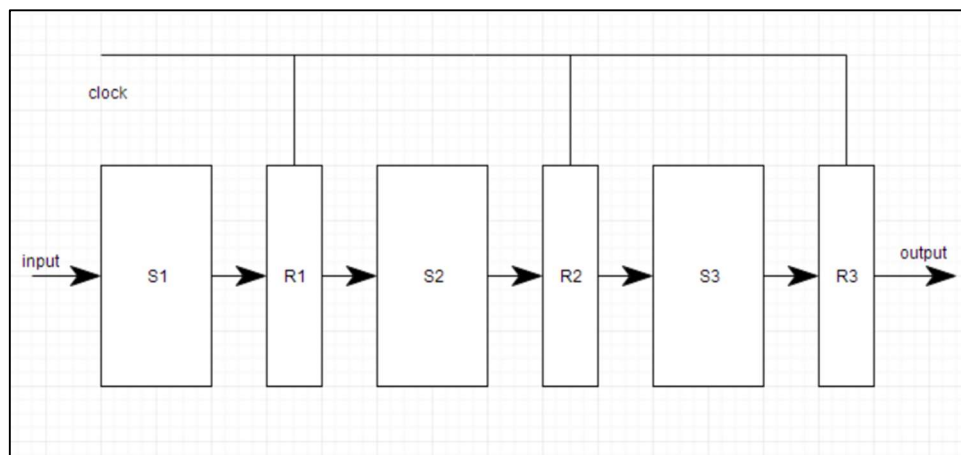
In manufacturing plants, as we know, assembly lines are irreplaceable in operations because they allow products at different stages of production to work at the same time. Pipelining is also used in computer architecture, where the previously approve input is displayed as output at the other end before new input is received [8].

We can divide instruction processing into two stages, fetching and executing instruction. In some cases, when we cannot access the main memory during instruction execution, it can be used to get the next instruction. Fetch instruction is the first stage, mainly for retrieving and buffering instruction and when the second stage is idle, the buffering instruction is transmitted to that stage [8]. When the executing instruction is executing in the second part, the first stage uses a never-used memory cycle to buffer and fetch the next instruction [9]. The method described above combines instruction buffering and requires the use of additional registers called instruction de-overlap or instruction prefetching.

The computer pipeline, it divided into several stages and in each stage connected to the next stage, thus forming a similar pipeline structure. The computer's instructions

come in one segment and output the result from the other segment, on the pipeline, it can be executed different parts of multiple instructions at the same time [10].

There are two main parts of the pipeline system. The first part is the register used to store general data, and the second is the combinational circuit which used to work on the register, while the output in the combinational circuit is used in the input register in the next section [10]. To brief, we use this figure to illustrate a simple pipeline system in computer architecture.



*Figure 3.1 - 1 Simple Pipeline in the Computer Architecture*

### 3.2 What is RISC (Reduced Instruction Set Computer)?

RISC uses microprocessors for information processing, and these microprocessors are supposed to perform tasks within the simplest instructions in the shortest possible time [11].

RISC took advantage of the fact that computers were built to process basic instructions quickly to develop microprocessors or chips and to speed up data processing, the theory is based on reducing the number of permanently stored instructions in the microprocessor and relying more on external instructions [11].

Originally, RISC microprocessors were used for large workstations or other high-end computer systems, but in the mid-1990s, RISC technology began to be integrated into personal computers, as we entered the 21<sup>st</sup> century, where technology grew, RISC was integrated into mobile devices such as smartphones and tablets [12].

### 3.3 Stages of RISC pipelining

#### Stage 1: Instruction fetches from memory (IF)

Get the current instruction from the memory and send the program counter (PC) to the memory. By adding 4 bytes to the PC, the PC is updated to the next consecutive PC [9].

#### Stage 2: Instruction decodes and register reads (ID)

The instruction is decoded and the register is read from the register corresponding to the register source specifier. For virtual branches, the equality test is performed when the register is read. Extend the instruction offset field if necessary. Multiply the added PC by the character extension offset to calculate the possible branch destination address [9].

Because registry descriptors are fixed in the RISC architecture, they can be decoded when the registry is read. Since the immediate part of the instruction is also in the same region, this technique is called constant field decoding. In the case where symbol extension real-time data is also evaluated in this loop [9].

#### Stage 3: Execute operation or calculate address (EX)

Depending on the type of instruction, the previous cycle provides the operands which are going to be operated one of three functions below by the ALU.

- Memory Reference: To create a valid address, the ALU adds a base address register and an offset.
- Register – Register ALU instruction: An ALU operates an operation defined by an ALU code for a value read from a register file.
- Register – Immediate ALU instruction: Read the first value from the register file and character extension, and ALU performs an operation supplied by the ALU code [13].

Since there are no instructions in the load and store architecture that need to compute both the data address and the operation data, the valid address and the execution cycle can be merged into one clock cycle [9].



#### Stage 4: Access memory operand (MEM)

Valid addresses and execution cycles can be combined into one clock cycle when there are no instructions in the load and store architecture that need to compute both the data address and the operation data [9].

#### Stage 5: Write result back to register (WB)

Register – Register ALU instruction or Load instruction:

The register file (for an ALU instruction) will be written with the result comes from the memory system or the ALU, [9].

### 3.4 Pipelining Hazard

A pipe hazard arises when a pipe or part of a pipe is forced to stop due to unforeseen circumstances. Resources, data, and control are the three common types of pipeline hazards Pipeline bubble is another term for pipe stagnation.

#### 3.4.1 Resources Hazards

Instructions must be executed consecutively, not in parallel, during a portion of the pipe. There is a resource risk when more than one instruction in the pipeline requests for the resource in the same position [14].

*Table 3.4 - 1 Example of resources hazards: conflict*

Instruction/ Cycle	1	2	3	4	5
I1	IF(Mem)	ID	EX	Mem	
I2		IF(Mem)	ID	EX	
I3			IF(Mem)	ID	EX
I4				IF(Mem)	ID

In the above situation, these two instructions are trying to access the same resource (memory) in cycle 4, which makes a conflict happen. In order to avoid this situation, the command should be configured to wait for the required resources to be available. Due to this delay, the pipeline will stop as follows:



*Table 3.4 - 2 Example of resource hazards:  
after setting the instruction to wait for accessibility.*

Instruction/ Cycle	1	2	3	4	5	6	7
<b>I1</b>	IF(Mem)	ID	EX	Mem	WB		
<b>I2</b>		IF(Mem)	ID	EX	Mem	WB	
<b>I3</b>			IF(Mem)	ID	EX	Mem	WB
<b>I4</b>				-	-	-	IF(Mem)

### 3.4.2 Data Hazards

Data is at risk when accessing operand entries causes a conflict. The hazard can be described in general terms as follow: in a program, the two instructions must be executed sequentially and both instructions must access the same register memory or operand. If you follow these two instructions exactly, you won't have any difficulty. However, if the instruction is executed in the pipeline, the operand's value can be modified, resulting in a different result than if the strict instruction was executed [15].

Types of Data Hazards:

- i. Write after reading (WAR):

It is a hazard to complete a written command before reading instruction in a register or memory location.

- ii. Read after write (RAW):

The registers or memory locations are modified by instructions can be a hazard if reads are completed before the writing process is complete. But there is no danger of data in register or memory location being reads by segmented instructions.

- iii. Write after write (WAW):

If both instructions are written in the same place, it can be dangerous to write in reverse order [8].

As an example, let's take the following sequence of instructions for an x86 computer.

```

ADD EAX, EBX /* EAX = EAX + EBX
SUB ECX, EAX /* ECX = ECX - EAX
    
```

Figure 3.4 - 1 Example of x86 machine instruction sequence [8].

The effect of writing EAX is to add the contents of the 32-bit EAX and EBX registers [8]. The second instruction stores the result of the ECX minus the EAX in the ECX register. The behavior of the pipeline is shown in Figures 3.4 – 2.

		Clock cycle									
		1	2	3	4	5	6	7	8	9	10
ADD EAX, EBX		FI	DI	FO	EI	WO					
SUB ECX, EAX			FI	DI	Idle		FO	EI	WO		
I3				FI			DI	FO	EI	WO	
I4							FI	DI	FO	EI	WO

Figure 3.4 - 2 Example of Data Hazard [8]

The EAX register will not be updated by the ADD instruction that occurs in the fifth clock cycle until the end of the step. The SUB command, on the other hand, needs a value at the beginning of the second stage, which occurs in another clock cycle. The pipeline must pause two clock cycles to continue running. In the absence of dedicated equipment and dedicated bypass methods, these data can lead to invalid pipeline accumulation [8].

### 3.4.3 Control Hazards

Control hazard usually happens when the pipeline misjudges the branch prediction and therefore must first delete the instruction before sending it to the pipeline.

This form of dependence occurs a control command is transmitted, and when new instructions are injected into the pipeline, the processor does not know the destination address of those instructions [16].

*Table 3.4 - 3 Sequence of instructions in the program [16]*

<b>Instruction/ Cycle</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>
<b>I1</b>	IF	ID	EX	Mem	WB	
<b>I2</b>		IF	ID(PC:250)	EX	Mem	WB
<b>I3</b>			IF	ID	EX	Mem
<b>BI1</b>				IF	ID	EX

**Output Result: I1 -> I2->I3->BI1**

The table above shows that the sequence output does not match the expectation, showing that the pipeline was not properly constructed. The only way to solve this problem is to prevent the instruction until we get the branch instruction destination address. This can be achieved by using a delay slot until the destination address is determined [16].

*Table 3.4 - 4 Sequence of instructions in the program after delay the instruction [16]*

<b>Instruction/ Cycle</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>
<b>I1</b>	IF	ID	EX	Mem	WB	
<b>I2</b>		IF	ID(PC:250)	EX	Mem	WB
<b>Delay</b>	-	-	-	-	-	-
<b>BI1</b>				IF	ID	EX

**Output Result: I1 -> I2 -> Delay (Stall) -> BI1**

This output sequence is the same as the predicted output sequence because the delay slot does not perform any operations [16]. However, this gap will stop the pipeline.

## Chapter 4 MIPS Development

### 4.1 Application of MIPS

MIPS, which means "microprocessor without internal interlocking pipeline stage", is a popular RISC processor in the world and its mechanism is aimed at using software methods to avert data-related problems in pipeline as much as possible. As one of the earliest commercial RISC architecture chips, MIPS adopts simplified instruction system computing structure to design chips. Compared with the Complex Instruction System Computing Architecture (CISC) adopted by Intel, RISC has the advantages of shorter development cycle, more convenient design and more advanced technologies, which is conducive to the development of a new generation of faster and stronger processors [17].

RISC platform was first born in MIPS host in 1980s. with the continuous development of related technologies, there have been a lot of corresponding application software, and its application fields have gradually expanded——MIPS processor can be seen from smart phones, microcontrollers to industrial control equipment. Chips based on this architecture are frequently used in many electronic products, personal equipment and commercial devices.

MIPS is the best RISC CPU sold in the company's products. We can see MIPS products are on sale from anywhere in related fields (such as Nintendo and Sony game consoles, Cisco routers and SGI supercomputers). R series microprocessor products of MIPS Chip Company are also adopted by many computer companies, which constitute various computer systems and workstations.

Compared with X86 and ARM architectures, MIPS has a lower licensing cost, so it is adopted by most chip manufacturers except Intel. From a more professional point of view, MIPS system structure is very advanced, and its instruction system has gone through general processor instruction systems MIPS I to MIPS V and the development of embedded instruction systems MIPS16, MIPS32 to MIPS64 has been very mature. Moreover, in the design concept of the chip, MIPS emphasizes that the software and hardware cooperate to improve the performance, while simplifying the hardware design as much as possible [18].

It is worth mentioning that MIPS instruction set architecture is developing rapidly. DSP module, multithreading module, SIMD module as well as virtualization module, these are adding on the basis of instruction. In line with the development trend

of applications, MIPS can keep up with the rapid changes of application requirements.

At present, the mainstream application of MIPS architecture is to provide kernel design (other companies carry out relevant design and production after MIPS authorization). MIPS has a wide range of applications, including low-end and high-end consumer products, communication products, network video, games and images.

The applications based on MIPS architecture had also made achievements at the enterprise level. In the field of consumer electronics, Broadcom, MTK and other companies use MIPS processor in home wireless router master chip and 4G LTE modem chip respectively. In the field of artificial intelligence, the embedded processor core of MIPS is applied by Wave Computing Company in AI hardware acceleration technology based on data flow architecture. The representative enterprise in IOT embedded MCU field is Microchip company from America. Unlike the ARM Cortex M-series processor core used in the development of most embedded microcontrollers in the world, the company has always adhered to the 32-bit embedded MIPS processor core.

## **4.2 Memory Organization in MIPS**

### ***4.2.1 Memory Organization in Computer Architecture***

Memory unit means a set of storage units. Usually, these units store information in binary code and they are written in bits.

Generally, two types of memory are widely used. The first one is volatile memory, which loses data when turning of power. Contrast to the volatile memory, the second type is non-volatile memory, which has the ability not to lose data when the power is cut down [19].

### ***4.2.2 Memory in MIPS***

Memory on MIPS is addressed in bytes, which means each memory address corresponds to an 8-bit value. The MIPS architecture can accommodate up to 32 address lines, resulting in a total of  $2^{32} \times 8$  memory, or 4 GB. There are  $2^{64}$  memory addresses in a 64-bit machine, total 16EB of memory.

There are 4 billion memory addresses in 8 bits in a 32-bit computer, while the register file is 32 bits wide and regarded to be 32 registers; the memory and register files are quite distinct. We can see from the diagram above that the memory is large and sluggish, while the register is small and fast. This will be a regular trade-off in this class.

The register file has 32-bit entries. So, to make 32 bits for the register file, we'll need four 8-bit locations to form the memory. It is known as a word in MIPS since it primarily deals with data in 4-byte chunks (32-bits) [20].

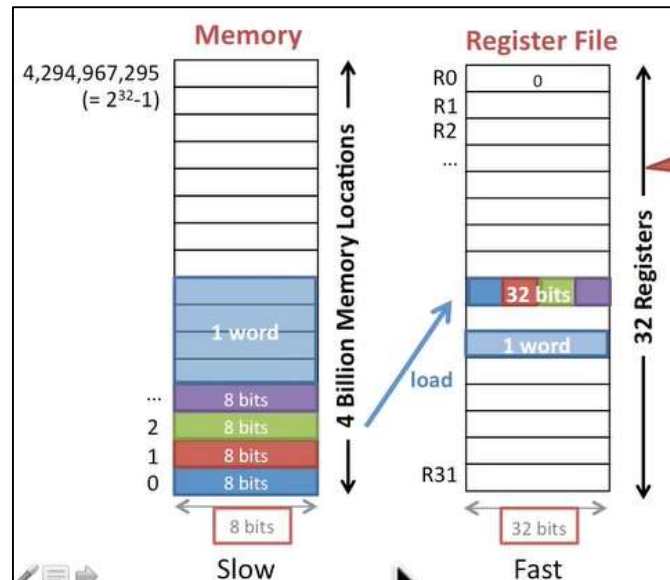


Figure 4.2—1 Memory vs Register

### 4.2.3 View memory as bytes or words

#### a. Byte-addressable

MIPS handled the most data in terms of words, but not bytes, thus we'll need to combine four pieces of memory to achieve the word's 32 bits. Figure 4.2 – 2 tells us the byte-addressable view of memory.

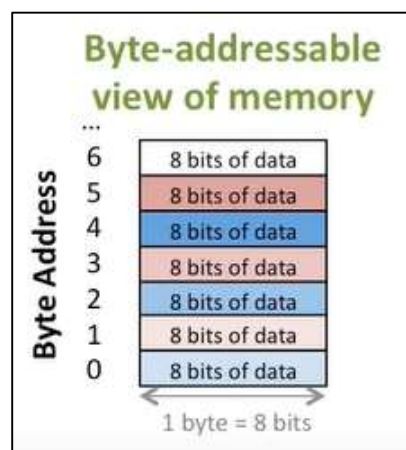


Figure 4.2 – 2 Byte-addressable

### b. Word-aligned

A register in the word-aligned mode may carry 32 bits of data, which is one word, and the addresses are the same as the registers, therefore the addresses can likewise hold 32 bits of data (1 word). The advantage of utilizing word-aligned is that we are able to load a word of data into the register when we load a word-aligned address, making it faster. We try to talk about the word-aligned view of memory in this figure.

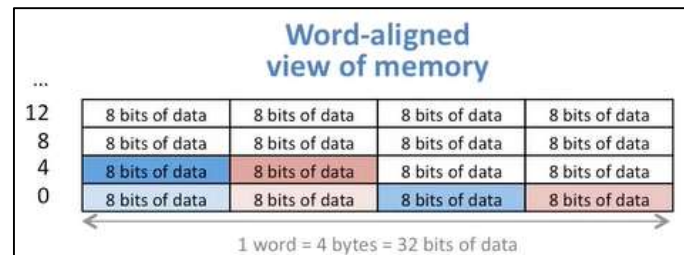


Figure 4.2 – 3 Word-aligned

## 4.3 Comparison of MIPS with other ISAs

Last but not least, it is the Comparison of MIPS with other ISAs. From the previous article, we can know that MIPS is a processor architecture based on RISC. However, there are some processor architectures based on CISC like x86. In order to obtain more objective and true results, we will compare 32-bit MIPS with 32-bit ARM and 32-bit x86 [21].

At first, we will compare their characteristics. For x86, forward compatibility is its biggest advantage. However, its instruction set addressing range is small, which restricts the needs of users, and the complex instructions make it difficult to design and manufacture processors. This is also the deficiency of the CISC system compared with the RISC system. Therefore, compared with x86, the instruction format of the ARM and MIPS architecture instruction sets are unified and there are fewer types, which improves the execution efficiency. However, MIPS has twice as many core registers as ARM, which can achieve higher performance and lower power consumption. And, MIPS has slightly more instructions than ARM, which makes it more flexible when performing some operations.

Then, we will discuss their software environments. x86 has the richest software environment. Windows, Linux and IOS can all run well on the x86 architecture hardware platform. The explosive development of the ARM platform originated from

the popularization of smart mobile devices. It is widely used in smart phones, tablet computers, etc., but it takes a long time to develop in the field of servers or computing. The MIPS architecture has long been more common in niche market products such as high-definition boxes, printers and network equipment, but compared to ARM, it has gained a certain opportunity in the server and computing fields [22].

Here, we draw a table to show the difference between these three architectures:

*Table 4.3 – 1 Comparison between ISAs*

	MIPS	ARM	x86
Based on	RISC	RISC	CISC
Software environments	HD box, printer, network equipment, router field etc.	Windows, Linux, mobile devices, hard disk etc. Widely used in embedded system design.	Windows, Linux and IOS etc.
Class of ISA	load-store	load-store	register-memory
Memory addressing	aligned	aligned	not aligned
Addressing modes	register, immediate and displacement	register, immediate and displacement and 3 different variants of displacement	immediate, displacement and PC-relative addressing
Types of sizes and operands	8,16,32,64 bits	8,16,32,64 bits	8,16,32,64, 80 bits
Control flow instructions	test the contents of registers	test condition code bits	test the contents of registers
Encoding an ISA	fixed length	fixed length	variable length
Characteristics	-It has a special divider which can execute division instruction. -It has a lot of registers. -The pipeline delay slot is visible. [23]	-Uniform instruction format, fewer types, and fewer addressing modes. -Has a streamlined instruction set. -Few branches, which enhance the code density and performance [24].	-Has a huge instruction set, as well as a variety of instruction types. -General-purpose registers are small. -Instruction set addressing range is small [22].



## Chapter 5 Conclusion

For the situation of MIPS today, one thousand people have one thousand views. But we can be sure that this is not a technical reason. We have always emphasized that in the early stage of entrepreneurship, the performance of MIPS has always been superior to ARM, but in the end ARM succeeded. This shows from the side that high performance does not mean you can succeed. It is the most important thing to be able to meet the needs of the market. This is the lesson MIPS has taught us [25].

Secondly, when you have many of the same competitors, the only things that can determine your win or loss are your business model, luck, and your corresponding ability to change according to changes in the end market. Back in 2007, if MIPS can quickly follow up to solve power consumption, improve the ecology, and promote cooperation with Fabless, there may be new variables in the current mobile processor landscape. On the contrary, look at ARM, the transformation of the Cortex-M series of licensing fees is a response to potential challenges. This response speed is not available in MIPS.

As a classic architecture, MIPS is well-designed and eye-catching. It is an example that many computer books often use when introducing architectures. However, in the engineering sector, MIPS is not very popular, and there are relatively few related application books and learning resources. On the contrary, ARM has done a better job in this regard. More participants have recently come, and there will be more sharing, which will attract more people to learn and continue to share more content. This virtuous circle is successful for any chip or open-source system. prerequisites.

Finally, due to the unshakable "historical position" of MIPS, a number of loyal users have been preserved. Up to now, some companies are still using the MIPS architecture to design processors and SoCs.

## Reference

- [1] Q. Zheng and X. Yang, "Brief analysis of MIPS assembly language and high-level language," *Journal of Capital Normal University(Natural Science Edition)*, pp. 29-31, 8 Oct 2009.
- [2] "High-Level Language," TechTerms, 12 May 2017. [Online]. Available: [https://techterms.com/definition/high-level\\_language](https://techterms.com/definition/high-level_language). [Accessed 25 May 2021].
- [3] "Machine Lanuage," TechTerms, 18 Jan 2019. [Online]. Available: [https://techterms.com/definition/machine\\_language#:~:text=Machine%20language%2C%20or%20machine%20code,they%20only%20recognize%20binary%20data..](https://techterms.com/definition/machine_language#:~:text=Machine%20language%2C%20or%20machine%20code,they%20only%20recognize%20binary%20data..) [Accessed 25 May 2021].
- [4] P. Prakash, "High level languages – advantages and disadvantages," Codeforwin, 17 May 2017. [Online]. Available: <https://codeforwin.org/2017/05/high-level-languages-advantages-disadvantages.html>. [Accessed 21 June 2021].
- [5] P. Pedamkar, "What is Assembly Language?," EDUCBA, [Online]. Available: <https://www.educba.com/what-is-assembly-language/>. [Accessed 21 June 2021].
- [6] G. d. R. Iván, M. H. Agustín, R. P. Óscar, J. A. Miguel, P. Pablo, d. S. Antonio, S. Jonatan 和 S. Sebastián, "A RISC-V Processor Design for Transparent Tracing," *Electronics*, 11 2020.
- [7] K. V. B. Y., D. Suman, G. Naina, B. Shivam, H. Y. Jawad, C. Anupam 和 M. Avi, "Towards Designing a Secure RISC-V System-on-Chip: ITUS," *Journal of Hardware and Systems Security*, 12 2020.
- [8] W. Stallings, Computer Organization and Architecture, 10 编辑, Pearson, 2016.
- [9] L. H. John 和 A. P. David, "Computer Architecture: A Quantitative Approach," *Appendix A*, pp. A1-A77, 2006.
- [10] S. Tonight, "What is Pipelining?," *Study tonight*, 2021.

- [11] T. E. o. E. Britannica, “RISC,” *Encyclopaedia Britannica*, 24 May 2021.
- [12] IBM100, “RISC Architecture” .*IBM100*.
- [13] A.-K. A. S. 和 O. S. S., “Hybrid branch prediction for pipelined MIPS processor,” *International Journal of Electrical and Computer Engineering (IJECE)*, 8 2020.
- [14] A. Hadizadeh 和 E. Tanghatari, “ Parallel Processor Architecture with a New Algorithm for Simultaneous Processing of MIPS-Based Series Instructions,” *Emerging Science Journal*, 12 2017.
- [15] “Will RISC-V revolutionize computing?,” *Communications of the ACM*, 04 2020.
- [16] GeeksforGeeks, “Computer Organization and Architecture | Pipelining | Set 2 (Dependencies and Data Hazard),” *GeeksforGeeks*, 31 May 2019.
- [17] H. Zhang, J. Zekun 和 L. Yong, “Design of a dual-issue RISC-V processor,” *Journal of Physics: Conference Series*, 12 2020.
- [18] H. Kaneko 和 A. Kanasugi, “An integrated machine code monitor for a RISC-V processor on an FPGA,” *Artificial Life and Robotics*, 03 2020.
- [19] s. tonight, “Memory Organization in Computer Architecture” .*study tonight*.
- [20] Y. B. Yao, X. B. Zeng 和 G. P. Zhao, “Design and Implementation of MIPS Simulator Oriented Memory Hierarchy Research” .*Applied Mechanics and Materials*.
- [21] W. Wenjuan, S. Dongchu, Y. Bo 和 L. Yong, “Intelligent Security Monitoring System Based on RISC-V SoC,” *Electronics*, 6 2021.
- [22] L. Shuai, "Analysis of X86, ARM, MIPS microprocessor architecture," *THE FORTUNE TIMES*, no. 12, 2015.

- [23] "MIPS," Wave Computing, [Online]. Available: <https://www.mips.com/products/architectures/>. [Accessed 5 July 2021].
- [24] Y.-Y. Chuang, "ARM Architecture," [Online]. Available: [https://www.csie.ntu.edu.tw/~cyy/courses/assembly/12fall/lectures/handouts/lec08\\_ARMarch.pdf](https://www.csie.ntu.edu.tw/~cyy/courses/assembly/12fall/lectures/handouts/lec08_ARMarch.pdf). [Accessed 5 July 2021].
- [25] Anonymous, "MIPS performance categories," *Medical Economics*, 5 2016.
- [26] F. M.-V. a. J. S. N. Jachimie, "CReconfigurable finite field instruction set architecture," 出处 *Proceedings of the 2007 ACM/SIGDA 15th international symposium on Field programmable gate arrays*.
- [27] E. A. Singh, "Instruction Cycle," University of Lucknow.
- [28] B. Agarwal, "Instruction Fetch Execute Cycle," 2004. [Online]. Available: <https://web.archive.org/web/20090611211308/http://www.cs.montana.edu/~bosky/cs518/ife/IFE.pdf>. [Accessed 25 May 2021].
- [29] Kondalalith1, "Pipelining vs Non-Pipelining," *GeeksforGeeks*, 30 April 2020.
- [30] G. S. N. Hill, "MIPS2020 wrap-up," *Yarmouth*, 4 2020.
- [31] K. Hodgson, "MIPS 2020 - A Gathering With Purpose," *Business And Economics--Marketing And Purchasing*, 4 2020.
- [32] R. A. Dowling, "Can you afford to avoid MIPS participation?," *Medical Sciences--Urology And Nephrology*, 2 2020.
- [33] T. Shryock, "A physician's challenge: Getting paid in 2020 and beyond," *Medical Sciences--Ophthalmology And Optometry*, 2 2020.
- [34] G. Beenen 和 S. Pichler, "A discussion forum on managerial interpersonal skills," *The Journal of Management Development; Bradford*, 2016.
- [35] S. Ives, "Overview of MIPS 2018," *Security Systems News*, 4 2020.

## Appendix 1 Contributions

Group Number and Student ID	Contributions to the Project
SWE2009510  Liu Aofan	<p>Task 1: Responsible for making charts and tables and write overview part 2</p> <p>Task 2: Revision Ang Chen Jia's part</p> <p>Task 3: Responsible for Abstract</p> <p>Task 4: Write meeting minutes 2</p>
SWE2009513  Su Yanyu	<p>Task 1: Write MIPS instruction set architecture</p> <p>Task 2: Revision Lai Renjie's part</p> <p>Task 3: Responsible for paper overall inspection</p> <p>Task 4: Write meeting minutes 5</p>
SWE2009689  Ang Chen Jia	<p>Task 1: Write pipelining in MIPS</p> <p>Task 2: Revision Liu Aofan's part</p> <p>Task 3: Responsible for polishing the paper</p> <p>Task 4: Write meeting minutes 1</p>
SWE2009507  Lai Renjie	<p>Task 1: Write overview 2 and making assignment thesis appendix</p> <p>Task 2: Revision Hu Xiniao's part</p> <p>Task 3: Responsible for research methodology</p> <p>Task 4:</p>

	Write meeting minutes 3
<p>SWE2009500</p> <p>Hu Xinao</p>	<p>Task 1: Write MIPS's memory organization, applications and make comparison between MIPS with other ISAs</p> <p>Task 2: Revision Su Yanyu's part</p> <p>Task 3: Responsible for paper conclusion</p> <p>Task 4: Write meeting minutes 4</p>

## Appendix 2 Record of meeting minutes

### SOF108 – Computer Architecture

#### Meeting Minute 1

**Date:** 9<sup>th</sup> May 2021

**Time:** 9:00pm

**Venue:** Teams

**Attendees:**

Liu Aofan	Su Yanyu
Hu Xinao	Lai Renjie
Ang Chen Jia	

**Prepared by:** Ang Chen Jia

No	Matters	Remarks						
1	<b>Formation of Group</b>  Leader: Liu Aofan Group members: 1. Su Yanyu 2. Hu Xinao 3. Lai Renjie 4. Ang Chen Jia							
2	<b>Meeting Minutes</b> <ul style="list-style-type: none"><li>- The meeting minutes will take turn to write.</li><li>- Estimate we will have at least 5 times of meeting throughout whole assignment.</li></ul>							
3	<b>Distribution of tasks</b> <ul style="list-style-type: none"><li>1. Summary, introduction and conclusion will do until we finalise the important contents.</li><li>2. Tasks</li></ul> <table><tr><th>Task</th><th>Responsible</th></tr><tr><td><b>Overview of computer organization and architecture</b><ul style="list-style-type: none"><li>- The basic instruction cycle</li><li>- High-level language, machine language, assembly language</li><li>- ISA</li></ul></td><td>Lai Renjie Liu Aofan</td></tr><tr><td><b>MIPS instruction set architecture</b><ul style="list-style-type: none"><li>- Data operations, data transfer operations, sequencing</li><li>- Instruction format, encoding, etc.</li></ul></td><td>Su Yanyu &amp; Assisted by Lai Renjie</td></tr></table>	Task	Responsible	<b>Overview of computer organization and architecture</b> <ul style="list-style-type: none"><li>- The basic instruction cycle</li><li>- High-level language, machine language, assembly language</li><li>- ISA</li></ul>	Lai Renjie Liu Aofan	<b>MIPS instruction set architecture</b> <ul style="list-style-type: none"><li>- Data operations, data transfer operations, sequencing</li><li>- Instruction format, encoding, etc.</li></ul>	Su Yanyu & Assisted by Lai Renjie	<p>The deadline to complete the first draft of assignment is on <b>22/5/2021 (Saturday)</b>.</p> <p>All group members can submit the complete work into the WeChat group earlier.</p>
Task	Responsible							
<b>Overview of computer organization and architecture</b> <ul style="list-style-type: none"><li>- The basic instruction cycle</li><li>- High-level language, machine language, assembly language</li><li>- ISA</li></ul>	Lai Renjie Liu Aofan							
<b>MIPS instruction set architecture</b> <ul style="list-style-type: none"><li>- Data operations, data transfer operations, sequencing</li><li>- Instruction format, encoding, etc.</li></ul>	Su Yanyu & Assisted by Lai Renjie							

	- Addressing modes in MIPS		
	<b>Pipelining in MIPS</b> - Five stages of MIPS pipeline data path	Ang Chen Jia	
	- <b>Memory organization in MIPS</b> - <b>Applications of MIPS</b> - <b>Comparison of MIPS with other ISAs</b>	Hu Xinao & Assisted by Liu Aofan	
<b>4</b>	<b>Next Meeting</b>  <b>Date: 23/5/2021 (Sunday)</b> <b>Time: TBC*</b> <b>Venue: Microsoft Teams</b>  *Will ask in group which time is prefer one day before the meeting.		



**SOF108 – Computer Architecture**

**Meeting Minute 2**

**Date:** 23<sup>th</sup> May 2021

**Time:** 8:00pm

**Venue:** Teams

**Attendees:**

Liu Aofan	Su Yanyu
Hu Xinao	Lai Renjie
Ang Chen Jia	

**Prepared by:** Liu Aofan

No	Matters	Remarks						
1	<b>Assignment Revision</b> 3. Give each other opinions on the parts they wrote. 4. Not everyone knows clearly about the format of IEEE. So, we talk about the problem of reference format. 5. Discussion title and paragraph format in the assignment paper. 6. Talk about the what we should write in the Conclusion and Abstract							
2	<b>Description of Coursework</b> - Study the Description of Coursework - Have a deeper understanding of the teacher’s requirements							
3	<b>Discussion on Revision tasks</b> 1. The quality of some pictures is blurred and the format of the picture serial number is incorrect. 2. Exchange to check each other’s grammar fault. 3. Check whether or not there is logical fault in the paper. 4. Some parts that are too large should be deleted, and some parts that are small should be added. 5. Revision tasks <table><tr><th>Revision Task</th><th>Responsible</th></tr><tr><td>Overview of computer organization and architecture part 1</td><td>Su Yanyu</td></tr><tr><td>Overview of computer organization and architecture part 2</td><td>Ang Chen Jia</td></tr></table>	Revision Task	Responsible	Overview of computer organization and architecture part 1	Su Yanyu	Overview of computer organization and architecture part 2	Ang Chen Jia	<p>The deadline to complete the first draft of assignment is on <b>07/06/2021 (Saturday)</b>.</p> <p>All group members can submit the complete work into the WeChat group earlier.</p>
Revision Task	Responsible							
Overview of computer organization and architecture part 1	Su Yanyu							
Overview of computer organization and architecture part 2	Ang Chen Jia							

	<b>MIPS instruction set architecture</b>	Hu Xinao	
	<b>Pipelining in MIPS</b>	Liu Aofan	
	<ul style="list-style-type: none"> <li>- <b>Memory organization in MIPS</b></li> <li>- <b>Applications of MIPS</b></li> <li>- <b>Comparison of MIPS with other ISAs</b></li> </ul>	Lai Renjie	
<b>4</b>	<b>Next Meeting</b>  <b>Date: 08/06/2021 (Tuesday)</b> <b>Time: TBC*</b> <b>Venue: Microsoft Teams</b>  *Will ask in group which time is prefer one day before the meeting.		

**SOF108 – Computer Architecture**

**Meeting Minute 3**

**Date: 8<sup>th</sup> June 2021**

**Time: 8:00pm**

**Venue: Microsoft Teams**

**Attendees:**

Liu Aofan	Su Yanyu
Hu Xinao	Lai Renjie
Ang Chen Jia	

**Prepared by: Lai Renjie**

No	Matters	Remarks
1	<b>Distribution of tasks</b>	<p>The deadline to complete the leftover work is after midterm.</p> <p>All group members can submit the complete work into the WeChat group earlier.</p>
	<b>Task</b>	
	<b>Research methodology</b>	
	<b>Abstract</b>	
	<b>Similar report 1</b>	
	<b>Similar report 2</b>	
	<b>Conclusion</b>	
2	<b>About duplicate checking</b> After the integration is completed, duplicate checking will be carried out first. If there is a large section of copy and paste, you can say it in advance and take it back for modification	
3	<b>Determine format</b> After voting, we chose indented.	
4	<b>Research and analysis topic</b>	
5	<b>Next Meeting</b>  <b>Date: 19/06/2021 (Saturday)</b> <b>Time: TBC*</b> <b>Venue: Microsoft Teams</b>  *Will ask in group which time is prefer one day before the meeting.	

**SOF108 – Computer Architecture**

**Meeting Minute 4**

**Date:** 17<sup>th</sup> June 2021

**Time:** 19:40

**Venue:** Wechat

**Attendees:**

Liu Aofan	Su Yanyu
Hu Xinao	Lai Renjie
Ang Chen Jia	

**Prepared by:** Hu Xinao

No	Matters	Remarks												
1.	<b>Contribution :</b> <ul style="list-style-type: none"><li>● Appendix Making and Overview Part1 Completed by Lai Renjie</li><li>● Graph/Tables Making and Overview Part2 Completed by Liu Aofan</li><li>● MIPS Instruction Set Architecture Completed by Su Yanyu</li><li>● Pipelining in MIPS Completed by Ang Chenjia</li><li>● MIPS Development Completed by Hu Xinao</li></ul>	<b>Revision Check:</b> <ul style="list-style-type: none"><li>● Checked by Su Yanyu</li><li>● Checked by Ang Chenjia</li><li>● Checked by Hu Xinao</li><li>● Checked by Liu Aofan</li><li>● Checked by Lai Renjie</li></ul>												
2.	<b>Meeting Minutes</b> <ul style="list-style-type: none"><li>- Summarize the completed tasks</li><li>- New tasks will be expected to complete on June 28</li></ul>													
3.	<b>Distribution of Revision Tasks</b> <table><tr><th>Revision Task</th><th>Responsible</th></tr><tr><td>Research methodology</td><td>Lai Renjie</td></tr><tr><td>Abstract</td><td>Liu Aofan</td></tr><tr><td>Similar report 1</td><td>Su Yanyu</td></tr><tr><td>Similar report 2</td><td>Ang Chen Jia</td></tr><tr><td>Conclusion</td><td>Hu Xinao</td></tr></table>	Revision Task	Responsible	Research methodology	Lai Renjie	Abstract	Liu Aofan	Similar report 1	Su Yanyu	Similar report 2	Ang Chen Jia	Conclusion	Hu Xinao	<p>The deadline to complete the leftover work is before the beginning of July.</p> <p>All group members can submit the complete work into the WeChat group earlier.</p>
Revision Task	Responsible													
Research methodology	Lai Renjie													
Abstract	Liu Aofan													
Similar report 1	Su Yanyu													
Similar report 2	Ang Chen Jia													
Conclusion	Hu Xinao													
4.	<b>Concerning to References</b>													
5.	<b>Everyone has comments on the group’s paper</b> <b>1. Parts to be deleted</b> <b>2. Parts to be added</b>													

**SOF108 – Computer Architecture**

**Meeting Minute 5**

**Date:** 3<sup>th</sup> July 2021

**Time:** 8:30pm

**Venue:** WeChat

**Attendees:**

Liu Aofan	Su Yanyu
Hu Xinao	Lai Renjie
Ang Chen Jia	

**Prepared by:** Su Yanyu

No	Matters	Remarks
<b>1</b>	<b>Report revision</b> 1. Correct grammar problems found after duplicate checking. 2. There are too few pictures in some parts of the report, so we decided to add some after discussion. 3. We made a contribution table according to everyone's work. (At the end of the report)	
<b>2</b>	<b>Report duplicates checking</b> After discussing and modifying the contents of the report, we submitted it to Turnitin for duplicate checking.	After a period of discussion and revision, the duplicate checking rate of the report decreased significantly.
<b>3</b>	<b>Collation of relevant data</b> We collated and summarized the materials and data used in the writing process.	
<b>4</b>	<b>Summary and reflection</b> At this point, the technical report of SOF108 has been basically completed. We have reflected on our work and attitude and gained quite a lot.	

## Appendix 3 Marking Rubrics

### APPENDIX 1

Component Title		A study on the MIPS Instruction Set Architecture (CLO 3 - Cognitive)				
Criteria		Score			Weight (%)	Marks
		Excellent	Average	Poor		
<b>Abstract, Introduction and Research Methodology</b> ▪ <i>Abstract is included</i> ▪ <i>A comprehensive overview of the topic and findings from the review.</i> ▪ <i>Clearly states objectives and scopes.</i> ▪ <i>Clearly explanation the methodologies</i>		2.5 ~ 3	1.5 ~ 2	0 ~ 1	3	
<b>Main content of the research</b> ▪ <i>Provide a comprehensive review of the MIPS architecture with adequate and suitable citations.</i> ▪ <i>The content is well organized and reflects in depth research.</i> ▪ <i>The report is rich with technical detail.</i> ▪ <i>Free from typos and grammatical errors.</i> ▪ <i>References are appropriate, in-text citations are provided.</i> <b>Conclusion</b> ▪ <i>Concludes the report with the findings and its implications</i> ▪ <i>Address the limitation of the study</i> ▪ <i>Suggest future directions for research</i>		6 ~ 7	4 ~ 5	1 ~ 3	7	
Component Title		A study on the MIPS Instruction Set Architecture (CLO4 - Teamwork)				
Criteria	Score and Descriptors			Weight (%)	Marks	
	Excellent (3)	Average (1.5 - 2)	Poor (0 - 1)			
<b>Roles &amp; Work Distribution</b>	The group establishes and <b>documents clear and formal roles</b> for each member and distributes the workload equally.	The group <b>establishes informal roles</b> for each member. The workload could be distributed more equally.	The group <b>does not establish roles</b> for each member and/or the workload is unequally distributed.	3		
Criteria	Score and Descriptors			Weight (%)	Marks	
	Excellent (2)	Average (1)	Poor (0)			
<b>Communication</b>	Everyone is <b>fully engaged</b> with effective exchange of ideas.	The group is <b>engaged but can be distracted</b> . Ideas are exchanged with encouragement.	The group is only <b>engaged with encouragement or not all members are engaged</b> . Ideas are not exchanged effectively	2		
TOTAL				15		

## Appendix 4 Digital Receipt




### Digital Receipt

This receipt acknowledges that Turnitin received your paper. Below you will find the receipt information regarding your submission.

The first page of your submissions is displayed below.

Submission author: Aofan Liu  
 Assignment title: A study on MIPS Instruction Set Architecture  
 Submission title: SOF108 Group Assignment  
 File name: V16.docx  
 File size: 2.14M  
 Page count: 47  
 Word count: 9,297  
 Character count: 48,551  
 Submission date: 06-Jul-2021 03:04AM (UTC+0800)  
 Submission ID: 1613744128

XIAMEN UNIVERSITY MALAYSIA		
		
Course Code:	SOF108	
Course Name:	Computer Architecture	
Lecturer:	Md Ibrahim Sultan	
Academic Session:	2021/04	
Assignment Title:	A study on the MIPS Instruction Set Architecture	
Submission Due Date:	06 July 2021	
Prepared by:	Student ID	Student Name
	SW12009510	Li Ai Aofan
	SW12009513	Ng Yee Yee
	SW12009549	Ang Chuan Jia
	SW12009597	Lim Ren Jie
	SW12009590	Hu Xian
Date submitted:	06 July 2021	
Component	Marks	Remark
A study on the MIPS Instruction Set Architecture		
Total	/15	
%	/15	

## Appendix 5 Similarity Report Summary

### SOF108 Group Assignment

#### ORIGINALITY REPORT

**10%**

SIMILARITY INDEX

**5%**

INTERNET SOURCES

**2%**

PUBLICATIONS

**8%**

STUDENT PAPERS

#### PRIMARY SOURCES

**1**

**Submitted to Xiamen University**

Student Paper

**7%**

**2**

**www.geeksforgeeks.org**

Internet Source

**<1%**

**3**

**"Fundamentals of Computer Organization and Design", Springer Science and Business Media LLC, 2003**

Publication

**<1%**

**4**

**wrap.warwick.ac.uk**

Internet Source

**<1%**

**5**

**"Informatics and Management Science II", Springer Science and Business Media LLC, 2013**

Publication

**<1%**

**6**

**D HARRIS. "Microarchitecture", Digital Design and Computer Architecture, 2007**

Publication

**<1%**

**7**

**lirias.kuleuven.be**

Internet Source

**<1%**

Submitted to CSU, San Jose State University



8	Student Paper	<1 %
9	Submitted to University of Pittsburgh Student Paper	<1 %
10	Submitted to Higher Education Commission Pakistan Student Paper	<1 %
11	Submitted to October University for Modern Sciences and Arts (MSA) Student Paper	<1 %
12	Submitted to Aston University Student Paper	<1 %
13	Submitted to Ibra College of Technology Student Paper	<1 %
14	repository.dl.itc.u-tokyo.ac.jp Internet Source	<1 %
15	doku.pub Internet Source	<1 %
16	www.assignmenthelp4me.com Internet Source	<1 %
17	Charles J. Sippl. "Macmillan Dictionary of Microcomputing", Springer Science and Business Media LLC, 1985 Publication	<1 %
18	en.wikibooks.org	

Internet Source

<1 %

19 epdf.pub  
Internet Source

<1 %

20 nsarchive.gwu.edu  
Internet Source

<1 %

21 Maya B. Gokhale, Judith D. Schlesinger.  
"Instruction Sets", Wiley, 2009  
Publication

<1 %

Exclude quotes On

Exclude matches < 3 words

Exclude bibliography On