指令集习题课

陈佳煜

2022.5.15

计算机指令系统复习

指令集重点内容

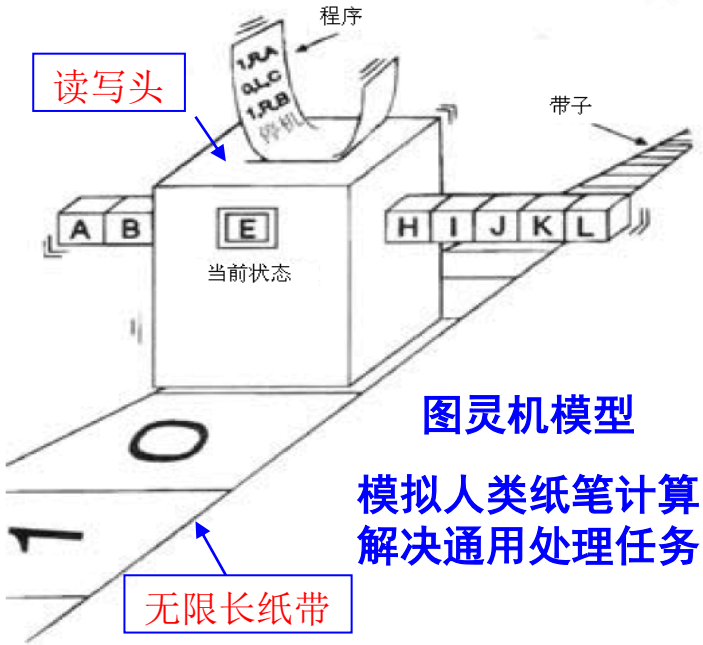
- **图灵机到通用计算机**
  - 图灵机定义
  - 通用计算机架构：冯诺依曼架构、哈佛架构
  - 指令集架构：RISC、CISC、URISC
- **MIPS指令集架构**
  - MIPS数据存储
  - 指令集格式：R型、I型、J型
  - 寻址方式：寄存器寻址、立即数寻址、基址或偏移寻址、PC相对寻址、伪直接寻址

指令集重点内容

- **MIPS指令系统**
  - 数据处理指令
  - 数据传送指令
  - 分支与跳转指令
- **汇编程序设计**
  - 基本程序结构
  - 变量与数组
  - 过程调用
- **CPU的性能评价**
  - 评价指标：延时、CPI
  - 影响CPU性能的因素
  - CPU性能的优化

图灵机

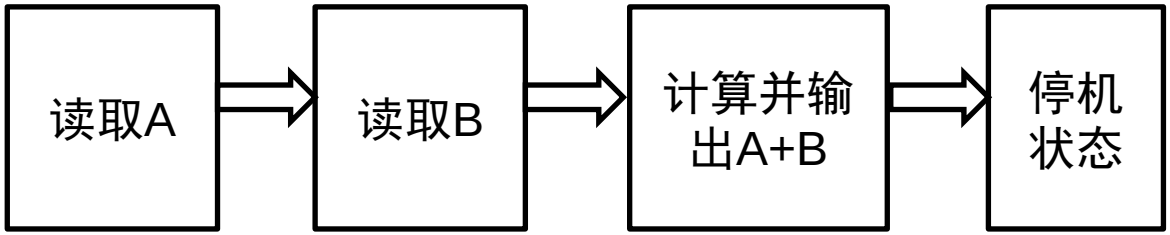
- 图灵机（Turing Machine）
  - 纸带+读写头+规则表+状态寄存器。
  - 以上部件，实际构成一有限状态机。
  - 图灵机根据当前状态和纸带上的符号，对照规则表决定如何进行读写头的移动与写入以及状态寄存器如何跳转。（思考：状态机是如何工作的）
  - 通常为了完成特定的任务，需要特定设计的规则表和对应的状态寄存器。



图灵机模型

模拟人类纸笔计算解决通用处理任务

一个计算A+B的例子

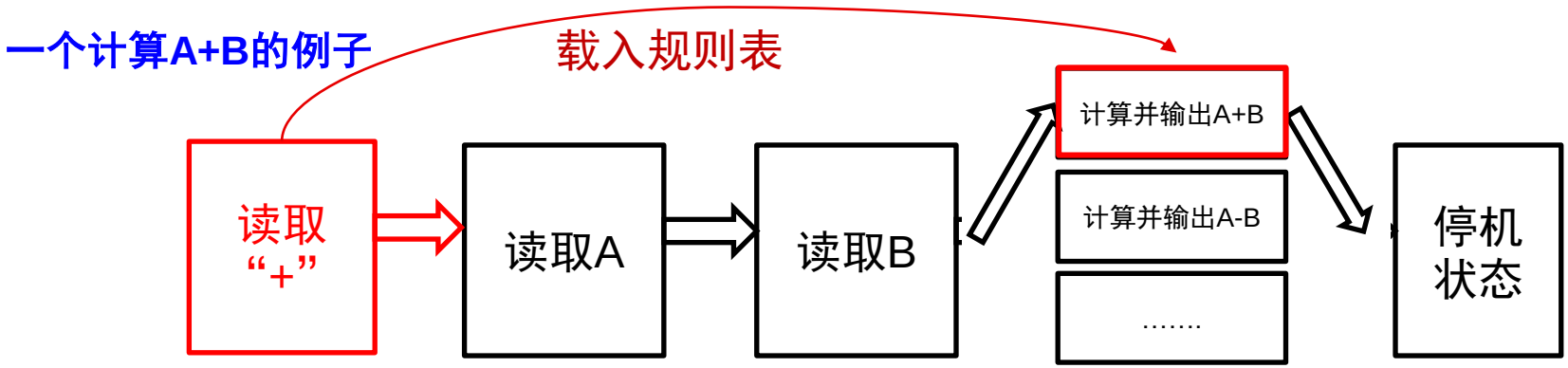


Turing A. On computable numbers, with an application to the Etscheidungs problem[J]. Proceedings of the London Mathematical Society, 1936, 42(1):230-265.

通用处理器的理论基础：图灵机

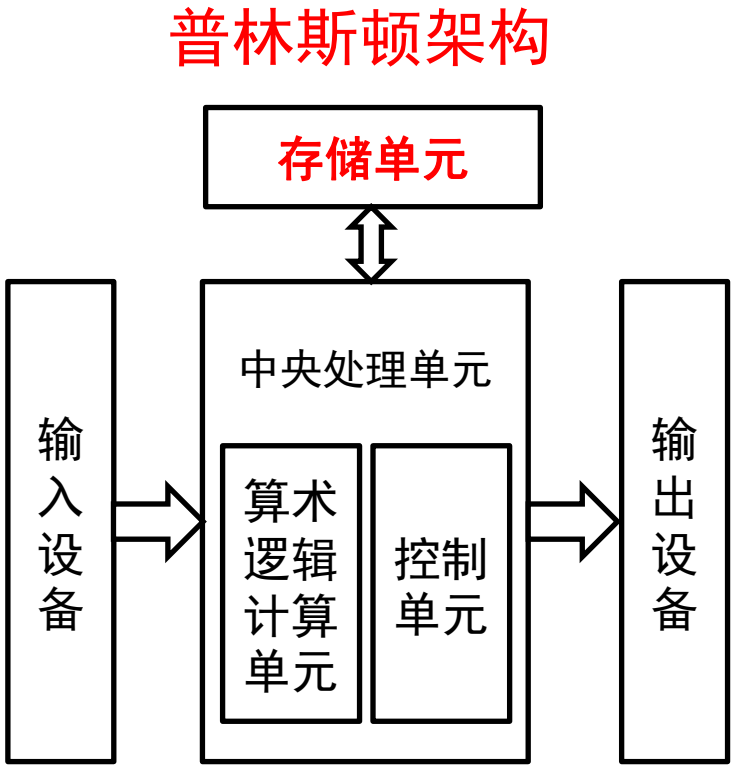
- 通用图灵机（Universal Turing Machine）
  - 图灵机根据规则表决定读写头的操作及状态跳转
  - 通用图灵机：将规则表作为一种输入，每个规则表是一种基础操作
  - 可以将复杂问题拆解为一系列基础操作 → 加载一系列规则表完成计算

读取纸带上的规则表，并根据规则表执行不同的计算过程



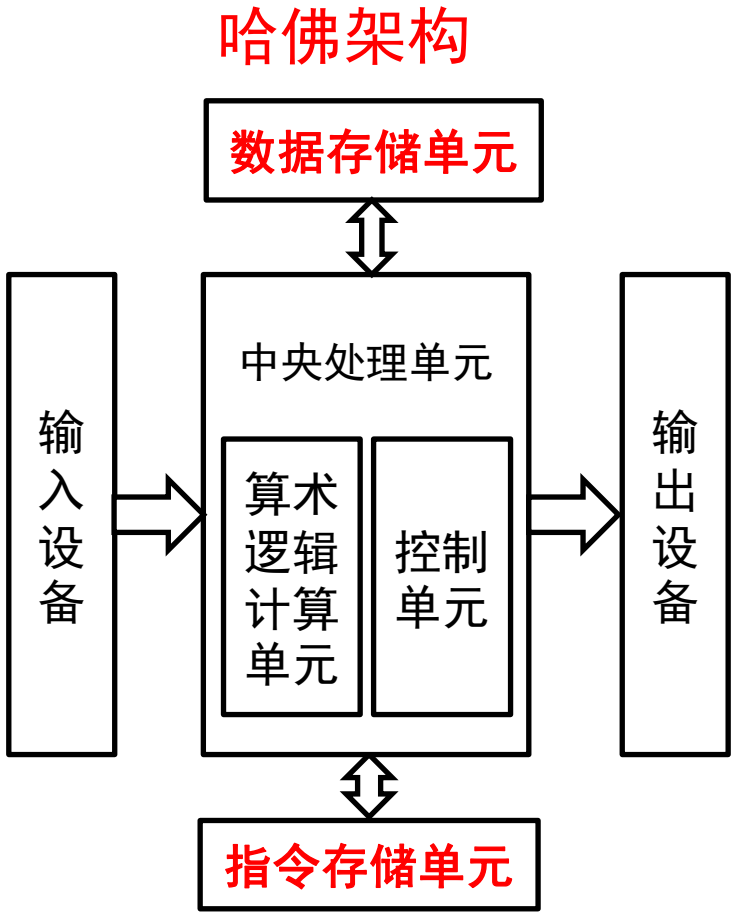
计算机：图灵机的具体实现

- 普林斯顿架构
  - 是一种图灵机的等效描述：
    - 包括中央处理单元（CPU），存储单元，输入设备与输出设备
    - 中央处理单元包括控制单元与运算单元
    - 程序作为一种数据顺序地存储于存储单元
  - 将程序作为一种数据存储起来的思想是计算机通用性的保证！



计算机：图灵机的具体实现

- 哈佛架构
  - 然而将程序与数据存储在同一个存储器中会降低系统的效率。在哈佛架构中，程序和数据存储在不同的存储器中，解决了这一问题。



指令集的分类

- 复杂指令集，CISC（Complex Instruction Set Computing）。指令种类多而复杂，每条指令字长不等，完成相同任务需要的指令数目少。代表：X86

- 精简指令集，RISC（reduced instruction set computing）。指令种类少，每条指令字长都相等，完成相同任务需要的指令数目多。代表：MIPS, RISC-V

- 最简指令集，URISC（ultimate reduced instruction set computer）。只有一条指令，但依然是完备的指令集。一般用来于教学和研究。

MIPS架构中的数据存储

- 指令集架构中数据应如何存储？

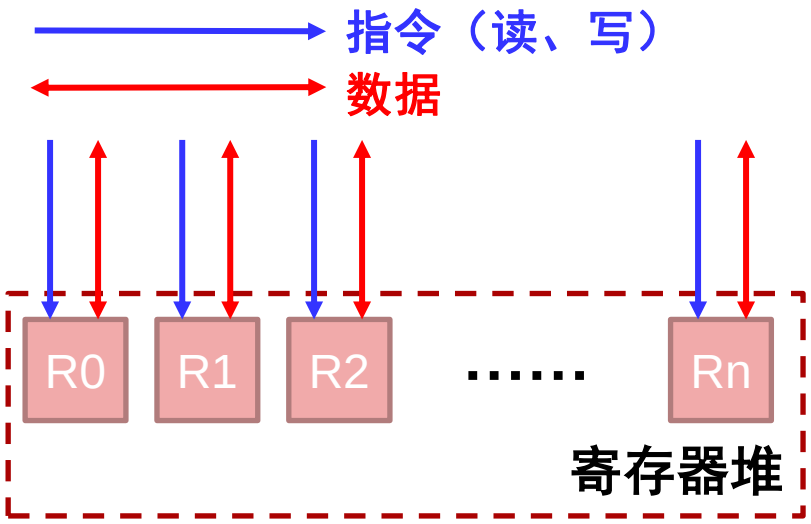
名称	实例	注释
32个寄存器	$s0, …$s7 $t0,…$t7	数据的快速定位，算术运算指令的操作数必须在寄存器中
2^{30}个存储字	存储器[1] … 存储器[2^{30}]	MIPS只能使用数据传送指令访问。MIPS中可以使用字(word)、半字(half word)、字节(byte)寻址，相邻数据字的地址相差**4(地址为字节地址)**。

MIPS中1word=4Byte=32bit

寄存器 vs. 存储器

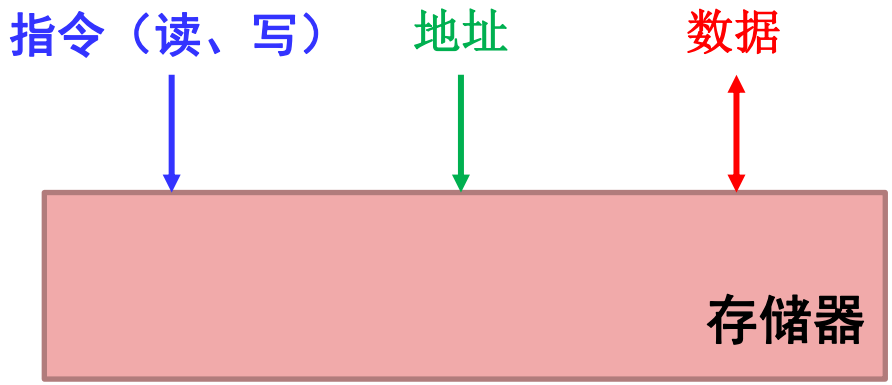
寄存器(ref:时序逻辑)

- 访问方式
  - 以寄存器编号进行访问，可同时访问不同寄存器
- 规模与速度
  - 一般个数少，访问速度快



存储器(ref:第九讲)

- 访问方式
  - 以存储器地址进行访问，不可同时访问不同地址
- 规模与速度
  - 一般地址范围大，访问速度慢



指令类型的判断
需要掌握

指令格式总结

- 具体指令含义后续会讲解，opcode/funct考试时会提供，无需记忆
- I型：用于有立即数的指令：lw sw beq bne
  - lw/sw: load word; store word
  - beq/bne: branch on equal; branch on not equal
  - opcode: 除000000, 00001x, 0100xx外
- J型：用于跳转，j, jal
  - j: jump; jal: jump and link;
  - opcode: 000010, 000011
- R型：所有其他指令
  - opcode: 000000, jr: jump register, srl, sll, sra

opcode: 区分不同指令的关键

指令和寻址方式
的对应自行复习！

MIPS寻址方式总结

- 寄存器寻址：找到对应的寄存器，从寄存器中取数/写数
- 立即数寻址：“访问”指令中的常数（立即数），无需访问存储器即可使用常数
- 基址或偏移寻址：指令中基址寄存器存储的基地址与指令中立即数的和是存储器中数据的地址
- PC相对寻址： PC_new = PC+4 + (Addr<<2)
- 伪直接寻址： PC_new = {PC[31:28], target_addr, 00}

汇编程序设计

- **汇编程序设计**
  - 基本程序结构
  - 变量与数组

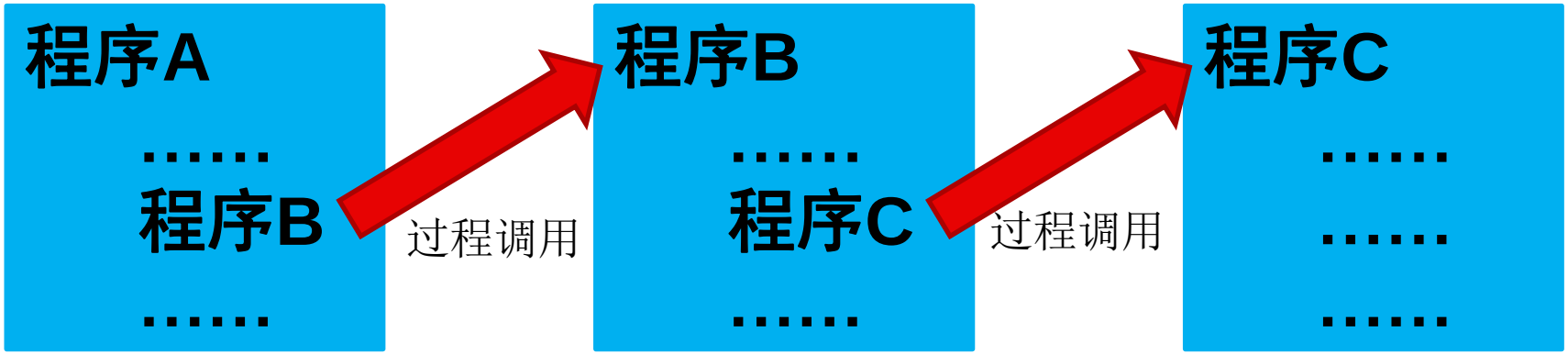
  根据代码判断功能，能够与C代码进行对应
  - 过程调用

  定义，栈的构成，实现方式

过程调用

一个过程包括入口，过程体和出口。

调用过程时需要准备好过程参数（如果有）并跳转到过程入口，在执行完过程体中的代码后从出口离开并回到主调过程的调用点的下一条语句，同时获得该过程的返回值（如果有）



CPU性能评价

- CPU执行时间 = CPU时钟周期数×时钟周期

$$= \frac{\text{CPU时钟周期数}}{\text{时钟频率}}$$

- CPU时钟周期数＝程序指令数×每条指令平均时钟周期数
- 每条指令平均时钟周期数——CPI(clock cycle per instruction)[6]

CPU执行时间 ＝ 指令数 × CPI × 时钟周期

$$CPI = \sum_{i=1}^{n} CPI_i \times P_i$$

影响性能的因素

- 影响计算机性能的因素

$$\text{CPU执行时间} = \text{指令数} \times \text{CPI} \times \text{时钟周期}$$

影响因素	影响
算法	指令数、CPI
程序设计语言	指令数、CPI
编译器	指令数、CPI
指令集体系架构	指令数、CPI、时钟周期
硬件实现	CPI、时钟周期

指令集第一次作业

CISC和RISC指令集的区别是什么？

CISC的计算机的指令系统比较丰富，有专用指令来完成特定的功能。因此，处理特殊任务效率较高，但是电路设计复杂。
而RISC的计算机指令简单，对不常用的功能，常通过组合指令来完成。因此实现特殊功能时，效率可能较低。但可以利用流水线技术和超标量技术加以改进和弥补。

（指令系统丰富or简单，处理特殊任务效率不同，电路复杂or简单，达出两点即可满分）

指令集第一次作业

对于32位MIPS而言，BEQ指令相对于给定的地址(二进制描述) 0x1234A000的跳转地址范围有多大，具体范围的上下界地址是多少（请注明单位，是字Word还是字节Byte）？如果想要前往该范围以外的地址，需要进行什么额外操作（说明一种可行方案即可）？

答案： 跳转范围： $-2^{15}\sim2^{15}-1$个**字** 或者 $-2^{15}+1\sim2^{15}$个**字**

上下界地址： $\boldsymbol{0x1232A004\sim0x1236A000}$

而BEQ是PC相对寻址，PC->PC+4+addr<<2
-2^{15}用二补码表示为1000 0000 0000 0000，移动两位做符号扩展为32位，0xFFFE0000
$2^{15}-1$表示为0111 1111 1111 1111，移动两位符号扩展为32位，0x0001FFFC

指令集第一次作业

对于32位MIPS而言，BEQ指令相对于给定的地址(二进制描述) 0x1234A000的跳转地址范围有多大，具体范围的上下界地址是多少（请注明单位，是字Word还是字节Byte）？如果想要前往该范围以外的地址，需要进行什么额外操作（说明一种可行方案即可）？

答案（续）：　使用 j 指令：

$beq\ \$s_0\ \$s_1\ L1$

$j\ L2$

$L1$: … …

$L2$: … …

j和jr有着更大的跳转范围

或者 jr 指令：将待跳转的地址存入寄存器

指令集第一次作业

请描述J型指令的格式，并说明J型指令跳转地址的范围。如果PC为0x005FCA90,请计算J指令的跳转范围（仅考虑理论范围）

答案： J型指令，6bit opcode + 26bit target addr

伪直接寻址：PC_new = {PC[31:28], target_addr, 00}

0000 00000000000000000000000000 00
0000 11111111111111111111111111 00

跳转范围$0x00000000 \sim 0x0FFFFFFC$

指令集第二次作业

1. CPU执行时间如何计算？

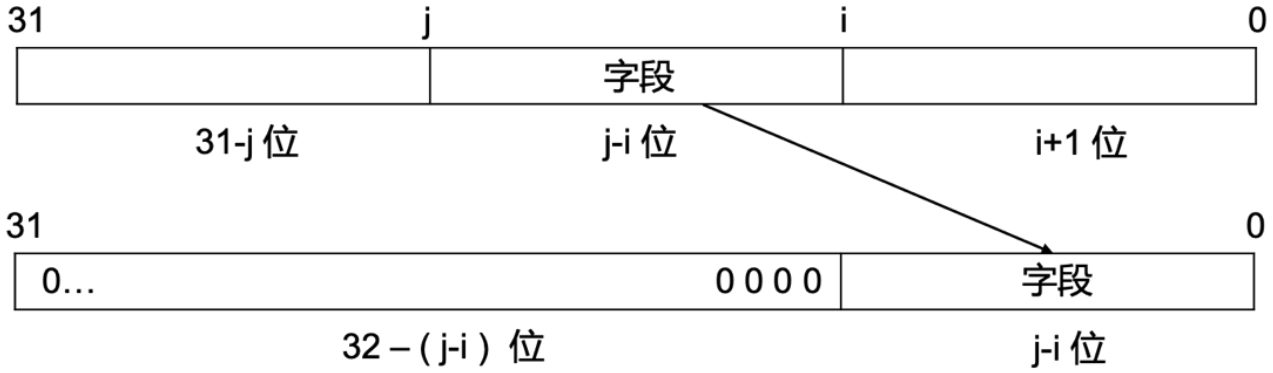
CPU执行时间 = 指令数 × CPI × 时钟周期

2.影响计算机系统性能的因素有哪些？分别是如何影响的？

影响因素	影响
算法	指令数、CPI
程序设计语言	指令数、CPI
编译器	指令数、CPI
指令集体系架构	指令数、CPI、时钟周期
硬件实现	CPI、时钟周期

指令集第二次作业

一些电脑有显式的指令从32位寄存器中取出任意字段并放在寄存器的最低有效位中，图 I-XXVI显示了需要的操作:



找出最短的MIPS指令序列能够在i=5和j=22的情况下从寄存器$t5中取出一个字段并放到寄存器$t0中。（提示：可以用两条指令实现）

答案：
```
sll $t0 $t5 9    #左移31-j位
srl $t0 $t0 15 #右移32-（j-i）位
```

左移时低位会补0，右移时高位会补0

指令集第二次作业

将下述代码在时钟频率为**2GHz**的机器上运行，各指令要求的周期数如下

指令	周期
add, addi, sll	1
lw,bne	2

$a2，$a3中的值均为2500．最坏情况下，将需要多少秒来执行下面这段代码

```
        sll   $a2，$a2，2
        sll   $a3，$a3，2
        add   $v0，$zero，$zero
        add   $t0，$zero，$zero
outer:  add   $t4，$a0，$t0
        lw    $t4，0($t4)
        add   $t1，$zero，$zero
inner:  add   $t3，$a1，$t1
        lw    $t3，0($t3)
        bne   $t3，$t4，skip
        addi  $v0，$v0，1
skip:   addi  $t1，$t1，4
        bne   $t1，$a3，inner
        addi  $t0，$t0，4
        bne   $t0，$a2，outer
```

指令集第二次作业

指令	周期
add, addi, sll	1
lw,bne	2

```
                1   sll   $a2，$a2，2
                1   sll   $a3，$a3，2
                1   add   $v0，$zero，$zero
                1   add   $t0，$zero，$zero
outer:  2500        add   $t4，$a0，$t0
        2500 *2     lw    $t4，0($t4)
        2500        add   $t1，$zero，$zero
inner:  2500*2500   add   $t3，$a1，$t1
        2500*2500*2 lw    $t3，0($t3)
        2500*2500 *2bne   $t3，$t4，skip
        2500*2500   addi  $v0，$v0，1
skip:   2500*2500   addi  $t1，$t1，4
        2500*2500 *2bne   $t1，$a3，inner
        2500        addi  $t0，$t0，4
        2500 *2     bne   $t0，$a2，outer
```

最坏情况对应于 inner 循环中 bne $t3, $t4, skip 语句每一次都不发生跳转，顺序执行 addi $v0, $v0, 1

因此总共执行指令数：

初始化：4 条算术指令

Outer 循环：一次 outer 循环执行 1 次完整的 inner 循环和额外的 3 条运算指令以及 2 条 lw/bne

Inner 循环：一次 inner 循环执行 3 条运算指令和 3 条 lw/bne

因此一次完整的 inner 循环执行 2500*3=7500 条运算指令和 7500 条 lw/bne

最终一共执行 (7500+3)*2500+4=18757504 条运算指令和 (7500+2)*2500=18755000 条 lw/bne

总周期数为 18757504*1+1875500*2=56267504

总共需要约 28.13ms

指令集第二次作业

16.有以下一段汇编程序和对应的 C 程序：

地址	汇编代码	注释	指令代号
0x00400000	addi $s0 $zero 21	int a = 21;	I1
0x00400004	addi $s1 $zero 0	int N = 0;	I2
0x00400008	while: slti $t0 $s1 1000	while 开始	I3
0x0040000c	addi $at $zero 1		I4
0x00400010	bne $t0 $at end		I5
0x00400014	andi $t0 $s0 1		I6
0x00400018	slti $t0 $t0 1		I7
0x0040001c	beq $t0 $zero else		I8
0x00400020			I9
0x00400024	j endif		I10
0x00400028	else: add $t0 $s0 $s0		I11
0x0040002c	add $t0 $t0 $s0		I12
0x00400030	addi $s0 $t0 1		I13
0x00400034	endif: slti $t0 $s0 2		I14
0x00400038	bne $t0 $zero end		I15
0x0040003c	addi $s1 $s1 1		I16

$s0代表a
$s1代表N

0x00400040	j while	while 结束	I17
0x00400044	end: addi $v0 $s1 0	设置返回值	I18

```
int a = 21;
int N = 0;
while(N <1000){
    if ((a&1)==0){
        a=a>>1;
    }
    else{
        a=a+a+a+1;
    }
    if (a<2) break;
    N=N+1;
}
```

指令集第二次作业

i. 请根据 C 语言代码写出汇编指令 I9，它的指令格式类型是什么（R，I，J）；

ii. I17 是 J 型指令，请写出该指令第 25-0 位（I17[25:0]）的值是多少，用 16 进制表示；

iii. 该程序执行结束时 I16 指令一共执行了多少次；

iv. 该汇编程序在一个主频为 1GHz 的单周期处理器上执行完成需要多少时间。

i. 对应 a=a>>1，**srl $s0, $s0, 1（或者 sra $s0, $s0, 1），R 型指令**

Shift and Rotate Operations		
ROTR[R2]	RD, RS, BITS5	RD = RS$_{BITS5-1:0}$ :: RS$_{31:BITS5}$
ROTRV[R2]	RD, RS, RT	RD = RS$_{RT4:0-1:0}$ :: RS$_{31:RT4:0}$
SLL	RD, RS, SHIFT5	RD = RS << SHIFT5
SLLV	RD, RS, RT	RD = RS << RT$_{4:0}$
SRA	RD, RS, SHIFT5	RD = RS$^{\pm}$ >> SHIFT5
SRAV	RD, RS, RT	RD = RS$^{\pm}$ >> RT$_{4:0}$
SRL	RD, RS, SHIFT5	RD = RS$^{\varnothing}$ >> SHIFT5
SRLV	RD, RS, RT	RD = RS$^{\varnothing}$ >> RT$_{4:0}$

Shift5代表的是R型指令中的移位量

ii. 对应 while 的地址，对比找到其中的 26 位。**0x100002**

0x00400008 伪直接寻址，PC[31:28] TargetAddr 00 0000 0000 0100 0000 | 0000 0000 0000 1000 ——> 0x100002

iii. 最终从 break 出，最后一次不执行，**执行了 6 次**

指令集第二次作业

16.有以下一段汇编程序和对应的 C 程序：

地址	汇编代码	注释	指令代号
0x00400000	addi $s0 $zero 21	int a = 21;	I1
0x00400004	addi $s1 $zero 0	int N = 0;	I2
0x00400008	while: slti $t0 $s1 1000	while 开始	I3
0x0040000c	addi $at $zero 1		I4
0x00400010	bne $t0 $at end		I5
0x00400014	andi $t0 $s0 1		I6
0x00400018	slti $t0 $t0 1		I7
0x0040001c	beq $t0 $zero else		I8
0x00400020	**srl $s0, $s0, 1**		I9
0x00400024	j endif		I10
0x00400028	else: add $t0 $s0 $s0		I11
0x0040002c	add $t0 $t0 $s0		I12
0x00400030	addi $s0 $t0 1		I13
0x00400034	endif: slti $t0 $s0 2		I14
0x00400038	bne $t0 $zero end		I15
0x0040003c	addi $s1 $s1 1		I16
0x00400040	j while	while 结束	I17
0x00400044	end: addi $v0 $s1 0	设置返回值	I18

执行流程：

循环外：I1+I2

If 判断执行 7 次：I3~I8

I9~I10 执行 6 次，第一次进 else 不执行

只有第一次执行 else：I11+I12+I13

执行 I14+I15 7 次

执行 I16+I17 6 次 最后一次不执行

只有最后一次执行 I18

共 2+6*7 + 2*6 + 3 + 2*7 + 2*6 + 1 = 86 个时钟周期

$$t = \frac{n}{f} = 86ns$$

指令集第二次作业

20. 某处理器的算术指令 CPI 为 1，Load/Store 指令 CPI 为 10，分支指令 CPI 为 3。假设一段程序有 800 万条算术指令，500 万条 Load/Store 指令和 100 万条分支指令。

i 计算该处理器运行这段程序的平均 CPI。

ii 假设我们能够为该处理器增加更加高效的指令，能够减少 20%的算术指令，但是会使得处理器的时钟频率降低为原来的 90%。从性能角度来说，我们是否应该增加这些指令？为什么？

iii 如果我们能够加速 Load/Store 指令至原来的 2 倍，则该处理器执行这段程序总的性能提升多少？加速 Load/Store 指令至原来的 10 倍呢？

（1）$\frac{800}{1400}*1+\frac{500}{1400}*10+\frac{100}{1400}*3=4.36$

（2）增加高效指令前$t=(800*1+500*10+100*3)*T=6100\ 万*T$

增加高效指令后$t'=(800*0.8*1+500*10+100*3)*\frac{T}{0.9}=6600\ 万*T$

所以不应该降低

（3）加速前$t=6100\ 万*T$

加速 2 倍后$t=\left(800*1+500*\frac{10}{2}+100*3\right)*T=3600\ 万*T$，性能提升为 1.6944 倍

加速 10 倍后$t=\left(800*1+500*\frac{10}{10}+100*3\right)*T=1600\ 万*T$，性能提升为 3.8125 倍