# 测试样例测试基础：

## 单周期 MIPS 处理器架构：

实验设计的内容在于正确实现各条具体的指令。在实现过程中基本上是一条指令进来后，开始处理，完成后既可以立即产生一个结果。并且在这个过程中硬件之间的差距被忽视，如：Memory 的读写与 CPU 单元的运算时间差。因此实验仿真的内容可以抽象为一个简单的单周期 MIPS 处理器架构，如 Figure 1。

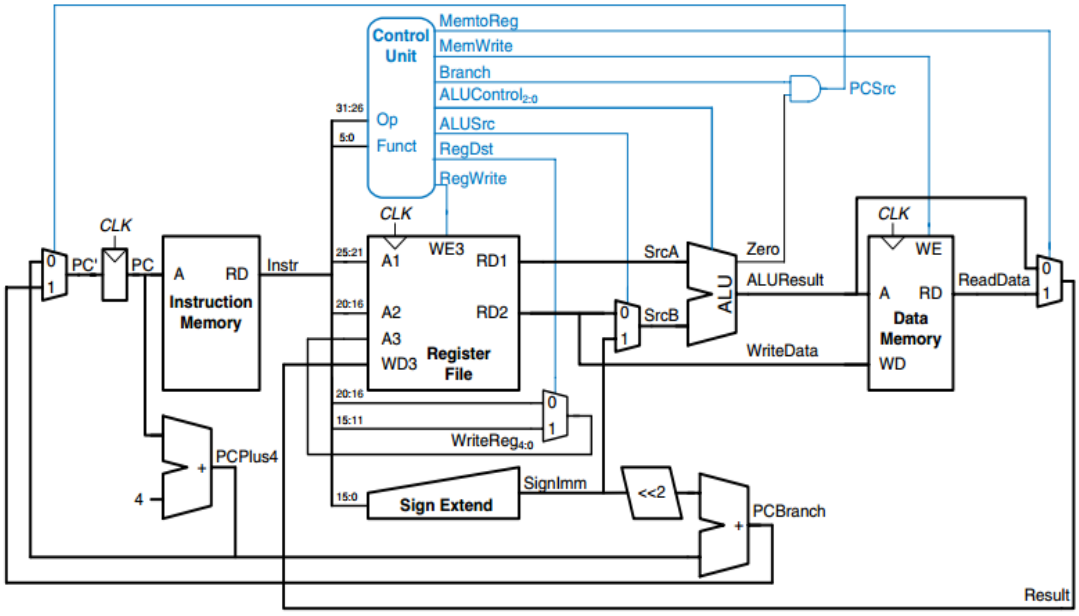


**Figure 7.11 Complete single-cycle MIPS processor**

Figure 1： A MIPS architecture From

在这一个架构中需要检测内容主要有决定 PC 的指令选址的选择即分支跳转指令，Register File 的正确读写，指令的解析，立即数的拓展，Memory 读写，ALU 运算单元的功能实现。测试样例的设计将基于此进行解析。

## 各功能单元的内容

1、 PC 的选取：决定计算机下一条指令是按正常顺序推进还是由计算机计算产生的特定位置偏移指令。

2、 Register File 的正确读写：寄存器列表中一共有 32 个寄存器，它们的功能与分布如 Figure 2 所示。其中可用于存储变量的寄存器为 t0~t9 和 s0~s7，共 18 个。值得注意的是 0 号寄存器单元是只读寄存器，并且存储值为常量 0。Register File 的正确读写旨在验证 0 号寄存器的可读不可写以及临时寄存器的正确使用。在本文测试中将 r8~r25 映射为 t0~t9 和 s0~s7 作为临时存储寄存器。

Table 6.1 **MIPS register set**

| Name | Number | Use |
|---|---|---|
| $0 | 0 | the constant value 0 |
| $at | 1 | assembler temporary |
| $v0-$v1 | 2–3 | procedure return values |
| $a0-$a3 | 4–7 | procedure arguments |
| $t0-$t7 | 8–15 | temporary variables |
| $s0-$s7 | 16–23 | saved variables |
| $t8-$t9 | 24–25 | temporary variables |
| $k0-$k1 | 26–27 | operating system (OS) temporaries |
| $gp | 28 | global pointer |
| $sp | 29 | stack pointer |
| $fp | 30 | frame pointer |
| $ra | 31 | procedure return address |

Figure 2：Registers Distribution List

3、 指令的解析

指令解析内容为能够正确识别汇编指令的类别（操作码）以及提取各个操作数。

4、 立即数的拓展

立即数的拓展为无法号拓展与有符号拓展。

5、 Memory 读写

内存设计中一个 Memory Cell 为 8bit，一个 cell 对应一个地址。

Instruction Memory 的读写，每条指令为 32bit 因此一条指令规定占用 4 个内存单元。

Data Memory 读写需要区分 Word，Half Word，Byte，它们分别为 32bit，16bit，8bit，各占用 4,2,1 个 Memory Cell。

6、 ALU 运算单元的功能实现

ALU 在 RV32I 的实现过程中主要包括加（+），减（-），与（&），或（|），异或（^），逻辑左移（<<<），逻辑右移（>>>），算术右移（>>）。

在验证过程中，各类指令都由各个组成单元组成，只要功能设计上没出现问题，架构结果一般不会出现问题，因此在测试过程中只需测试各类指令，不需要每条指令挨个测试。

# 测试样例

本文的测试采用汇编程序段的方式进行测试，可使用的寄存器为 0 号单元寄存器：x0，和临时寄存器 t0~t9 和 s0~s7：r8~r25。采用四个程序分别验证，包含有加法器、排序、最大公约数与最小公倍数和逻辑运算的最小值计算。对于每个验证程序提供 C++翻译代码用于验证结果，可以直观的观察运行结果是否正确。如不正确 C++结果将与汇编程序运行结果有差异。接下来将会对四个程序段进行描述。具体指令解析详见对应的 C++程序，在对应文件夹下。

```
1   LW r17,4(x0)
2   LW r18,8(x0)
3   ADDI r19,x0,0
4   ADDI r20,x0,0
5   ADDI r21,x0,0
6   XOR r19,r17,r18
7   AND r20,r17,r18
8   SLLI r20,r20,1
9   BEQ r20,x0,24
10  ADDI r21,r19,0
11  XOR r19,r19,r20
12  AND r20,r21,r20
13  SLLI r20,r20,1
14  BNQ r20,x0,-16
15  SW r19,12(x0)
16  ADDI x0,x0,1
```

Figure 3：Adder

```
1   LW r19,0(x0)
2   ADDI r19,r19,-2
3   ADD r17,x0,x0
4   SLLI r18,r19,2
5   LW r20,4(r18)
6   LW r21,8(r18)
7   BLT r21,r20,12
8   SW r20,8(r18)
9   SW r21,4(r18)
10  ADDI r18,r18,-4
11  BGE r18,r17,-24
12  ADDI r17,r17,1
13  BGE r19,r17,-36
14  ADDI x0,x0,0
```

Figure 4：Bubble sort

```
1   LW r17,4(x0)
2   LW r18,8(x0)
3   ADDI r19,r17,0
4   SUB r17,r17,r18
5   BGE r17,x0,12
6   ADDI r17,r18,0
7   ADDI r18,r19,0
8   BEQ r17,x0,8
9   JAL x0,-24
10  SW r18,12(x0)
11  LW r17,4(x0)
12  LW r18,8(x0)
13  ADDI r21,x0,0
14  ADDI r20,x0,0
15  ADDI r20,r20,1
16  SUB r18,r18,r19
17  BEQ r18,x0,8
18  JAL x0,-12
19  ADDI r20,r20,-1
20  BLT r20,x0,12
21  ADD r21,r21,r17
22  JAL x0,-12
23  SW r21,16(x0)
24  ADDI x0,x0,1
```

Figure 5：GCD_LCM

```
1   LW r16,0(x0)
2   LW r17,4(x0)
3   ADDI r25,x0,8
4   AND r18,r16,r17
5   SW r18,0(r25)
6   OR r18,r16,r17
7   SW r18,4(r25)
8   XOR r18,r16,r17
9   SW r18,8(r25)
10  SRA r18,r17,r16
11  SW r18,12(r25)
12  SLL r18,r17,r16
13  SW r18,16(r25)
14  SRL r18,r17,r16
15  SW r18,20(r25)
16  LW r18,28(x0)
17  ADDI r20,x0,6
18  ADDI r21,x0,2
19  SLLI r19,r20,2
20  LW r22,0(r19)
21  SLTU r23,r22,r18
22  BNQ r23,x0,8
23  JALR x0,x0,12
24  ADDI r18,r22,0
25  JAL x0,8
26  ADDI r18,r18,0
27  ADDI r20,r20,-1
28  BGE r20,r21,-36
29  SW r18,24(r25)
30  ADDI x0,x0,0
```

Figure 6：MinResult

# 加法器 —— Adder

如图 Figure 3，实现内容为：32 位 Integer 的加法器，通过位运算实现两个 Integer 的相加求和。涵盖指令功能 Load、Store、Immediately、Plus、Exclusive or、And、Logical left shift，包含 branch 指令：BEQ、BNQ。需要注意的有负数移位过程中回抵达 Integer 的下边界，以及

位操作处理过程中需要使用补码运算。

## 冒泡排序 —— Bubble Sort

如图 Figure 4，实现内容为：输入一个数组的长度，并输入这个长度的 Integer，通过冒泡排序对这个数组进行从大到小排序。涵盖指令功能 Load、Store、Immediately、Plus、Logical left shift。包含分支跳转指令：BLT、BGE。其中在这里的移位指令“SLLI r18,19,2”是为了保证内存读取 32 位数据是的地址格式对齐。需要注意的有 32 位 Integer 内存读写过程中单元地址对齐问题。

## 最大公约数最小公倍数 —— GCD_LCM

如图 Figure 5，实现内容为：给定两个 Integer，计算出这两个数的最大公约数与最小公倍数。涵盖指令功能 Load、Store、Immediately、Plus、Subtract。包含分支跳转指令：BGE、BEQ、JAL。需要注意的 0 号单元寄存器是 JAL 的目的寄存器，如果在这里没有对它进行只读处理将可能会导致后续指令执行出错。

## 最小逻辑运算结果 —— MinResult

如图 Figure 6，实现内容为：给定两个 Integer，计算出这两个数的通过位操作与、或、非、异或、算术右移、逻辑左移、逻辑右移的值，并求出它们的最小值。涵盖指令功能 Load、Store、Immediately、Plus、And、Or、Xor、Arithmetic right shift、Logical left shift、Logical right shift、Unsigned compare。包含分支跳转指令：BGE、BNQ、JALR。需要注意的是算术移位与逻辑移位的差异性。

上述测试样例涵盖了 MIPS 单周期架构的各个功能部件的应用，并且提供了 C++程序，结果可以直观的看出。但这几个测试样例没有涵盖所有指令。若有必要可以进行拓展，对其余指令进行逐条测试，生成结果输出。

Note:
All of architecture figures are from “Digital Design and Computer Architecture” by David Money Harris &Sarah L. Harris