计算机组成原理　第三周作业　3 月 6 日 周五

PB18151866 龚小航

2.10.

以下问题是关于将表中表示指令操作的位条目转换成汇编代码并且指出每条 MIPS 指令的格式

a.	1010 1110 0000 1011 0000 0000 0000 0100
b.	1000 1101 0000 1000 0000 0000 0100 0000

2. 10. 1 [5] < 2.5 >求上面的位条目分表表示什么指令。

2. 10. 2 [5] < 2.5 >求上面的位条目所表示的指令的类型(I 型或 R 型)。

2. 10. 3 [5] < 2.5 >如果上面的位条目是数据，求上面数字的十六进制表示。

以下问题是关于将表中的 MIPS 指令转换成位模式并且指出每条 MIPS 指令的格式类型。

a.	$add \quad \$t0, \$t0, \$zero$
b.	$lw \quad \$t1,\ 4\ (\$s3)$

2. 10. 4 [5] < 2.4 2.5 >上表中指令的十六进制表示。

2. 10. 5 [5] < 2.5 >求上面指令的类型(I 型或 R 型)。

2. 10. 6 [5] < 2.5 >求指令的十六进制表示的$opcode$、s和rt字段；

对于R型指令，求十六进制表示的rd和$funct$字段；

对于I型指令，求十六进制表示的立即数字段。

解：(1) 将指令写成标准格式：

	op	rs	rt	$address$	指令格式
a.	101011	10000	01011	00000　　00000　　000100	
	$(43)_{10}: sw$	$(16)_{10}: \$s0$	$(11)_{10}: \$t3$	$(4)_{10}$：立即数	I 型
b.	100011	01000	01000	00000　　00001　　000000	
	$(35)_{10}: lw$	$(8)_{10}: \$t0$	$(8)_{10}: \$t0$	$(64)_{10}$：立即数	I 型

因此易写出这两条指令的汇编形式：

$a. \quad sw \quad \$t3\,, \quad 4(\$s0)$

$b. \quad lw \quad \$t0\,, \quad 64(\$t0)$

(2) 这两条指令均为I型指令。

(3) 直接将四位二进制数组成一组，对应一个十六进制数。将 32 位二进制数转换，得到：

$a.\ (1010\ 1110\ 0000\ 1011\ 0000\ 0000\ 0000\ 0100)_B = (AE0B0004)_H$

$b.\ (1000\ 1101\ 0000\ 1000\ 0000\ 0000\ 0100\ 0000)_B = (8D080040)_H$

(4) 将指令列表，把每一项转换成二进制指令：

	op	rs	rt	rd	$shamt$	$funct$
a.	add	$\$t0$	$\$zero$	$\$t0$	0	$(32)_{10}$
R 型	000000	01000	00000	01000	00000	100000
	op	rs	rt	$address$		
b.	lw	$\$s3$	$\$t1$	$(4)_{10}$		
I 型	100011	10011	01001	00000	00000	000100

所以，这两条指令的十六进制表示为：

$a.\ (000000\ 01000\ 00000\ 01000\ 00000\ 100000)_B = (01004020)_H$

$b.\ (100011\ 10011\ 01001\ 00000\ 00000\ 000100)_B = (8E690004)_H$

(5) a指令为R型指令，b指令为I型指令。

(6) 上表已经完全列出每一项所对应的值：

$a.$ op = 0x0 ;　rs = 0x8 ;　rt = 0x0 ;　rd = 0x8 ;　funct = 0x20

$b.$ op = 0x23 ;　rs = 0x13 ;　rt = 0x9 ;　address = 0x

2.13.

在下面的问题中，数据表中包含寄存器$t0和$t1的值。按照下面的逻辑指令对这些寄存器进行操作。

a.	$t0 = 0x 5555 5555, $t1 = 0x 1234 5678
b.	$t0 = 0x BEAD FEED, $t1 = 0x DEAD FADE

2.13.1 [5] < 2.6> 求执行下面的指令序列后寄存器$t2的值。

```
sll  $t2, $t0, 4
or   $t2, $t2, $t1
```

2.13.2 [5] < 2.6 > 求执行下面的指令序列后寄存器$t2的值。

```
sll  $t2, $t0, 4
andi $t2, $t2, -1
```

2.13.3 [5] < 2.6 > 求执行下面的指令序列后寄存器$t2的值。

```
srl  $t2, $t0, 3
andi $t2, $t2, 0xFFEF
```

在下面的练习中，数据表中包含不同的 MIPS 逻辑操作，对于给定的不同寄存器$t0和$t1值，求这些操作最终的结果。

a.	*sll*	$t2, $t0, 1	*b.*	*srl*	$t2, $t0, 1
	or	$t2, $t2, $t1		*andi*	$t2, $t2, 0x00F0

2.13.4 [5] < 2.6 > 假设 $t0 = 0x 0000A5A5, $t1 = 0x 00005A5A 求执行上表中的指令后寄存器$t2中的值。

2.13.5 [5] < 2.6 > 假设 $t0 = 0x A5A5 0000, $t1 = 0x A5A5 0000 求最终寄存器$t2中的值。

2.13.6 [5] < 2.6 > 假设 $t0 = 0x A5A5 FFFF, $t1 = 0x A5A5 FFFF 求最终寄存器$t2中的值。

解：(1) 这两条指令的含义是：$t2 = $t0 << 4；$t2 = $t2 | $t1. 写出每步操作后的结果：

① 对于*a.*情况中寄存器的值：经过第一步操作后，寄存器内的值为：

$t0 = 0x 5555 5555, $t1 = 0x 1234 5678, $t2 = 0x 5555 5550

再将两个寄存器中的内容进行按位或：

$$\$t2 = \begin{array}{l} 0101\ 0101\ 0101\ 0101\ 0101\ 0101\ 0101\ 0000 \\ 0001\ 0010\ 0011\ 0100\ 0101\ 0110\ 0111\ 1000 \end{array} | \\ = 0101\ 0111\ 0111\ 0101\ 0101\ 0111\ 0111\ 1000 = 0x\ 5775\ 5778$$

② 对于*b.*情况中寄存器的值：经过第一步操作后，寄存器内的值为：

$t0 = 0x BEAD FEED, $t1 = 0x DEAD FADE, $t2 = 0x EADF EED0

再将两个寄存器中的内容进行按位或：

$$\$t2 = \begin{array}{l} 1110\ 1010\ 1101\ 1111\ 1110\ 1110\ 1101\ 0000 \\ 1101\ 1110\ 1010\ 1101\ 1111\ 1010\ 1101\ 1110 \end{array} | \\ = 1111\ 1110\ 1111\ 1111\ 1111\ 1110\ 1101\ 1110 = 0x\ FEFF\ FEDE$$

(2) 这两条指令的含义是：$t2 = $t0 << 4；$t2 = $t2 &(−1). 写出每步操作后的结果：

① 对于*a.*情况中寄存器的值：经过第一步操作后，寄存器内的值为：

$t0 = 0x 5555 5555, $t1 = 0x 1234 5678, $t2 = 0x 5555 5550

再将$t2寄存器内的值与−1按位与：

$$\$t2 = \begin{array}{l} 0101\ 0101\ 0101\ 0101\ 0101\ 0101\ 0101\ 0000 \\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111 \end{array} \& \\ = 0101\ 0101\ 0101\ 0101\ 0101\ 0101\ 0101\ 0000 = 0x\ 5555\ 5550$$

② 对于*b.*情况中寄存器的值：经过第一步操作后，寄存器内的值为：

$t0 = 0x BEAD FEED, $t1 = 0x DEAD FADE, $t2 = 0x EADF EED0

再将$t2寄存器内的值与−1按位与：

$$\$t2 = \begin{array}{l} 1110\ 1010\ 1101\ 1111\ 1110\ 1110\ 1101\ 0000 \\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111 \end{array} \& \\ = 1110\ 1010\ 1101\ 1111\ 1110\ 1110\ 1101\ 0000 = 0x\ EADF\ EED0$$

(3) 这两条指令的含义是：$t2 = $t0 >> 3；$t2 = $t2 &(0xFFEF). 写出每步操作后的结果：

① 对于*a.*情况中寄存器的值：经过第一步操作后，寄存器内的值为：

$t0 = 0x 5555 5555, $t1 = 0x 1234 5678,

$t2 = (0101 0101 0101 0101 0101 0101 0101 0101) >> 3

= (0000 1010 1010 1010 1010 1010 1010 1010) = 0x 0AAA AAAA

再将$t2寄存器内的值与0xFFEF按位与：

$$\$t2 = \begin{array}{l} 0000\ 1010\ 1010\ 1010\ 1010\ 1010\ 1010\ 1010 \\ 0000\ 0000\ 0000\ 0000\ 1111\ 1111\ 1110\ 1111 \end{array} \& \\ = 0000\ 0000\ 0000\ 0000\ 1010\ 1010\ 1010\ 1010 = 0x\ 0000\ AAAA$$

② 对于*b.*情况中寄存器的值：经过第一步操作后，寄存器内的值为：

$t0 = 0x BEAD FEED, $t1 = 0x DEAD FADE,

$t2 = (1011 1110 1010 1101 1111 1110 1110 1101) >> 3

= (0001 0111 1101 0101 1011 1111 1101 1101) = 0x 17D5 BFDD

再将$t2寄存器内的值与0xFFEF按位与：

$$\$t2 = \begin{array}{l} 0001\ 0111\ 1101\ 0101\ 1011\ 1111\ 1101\ 1101 \\ 0000\ 0000\ 0000\ 0000\ 1111\ 1111\ 1110\ 1111 \end{array} \& \\ = 0000\ 0000\ 0000\ 0000\ 1011\ 1111\ 1100\ 1101 = 0x\ 0000\ BFCD$$

(4) $t0 = 0x 0000A5A5, $t1 = 0x 00005A5A

① 对于*a.*情况中的指令含义为：$t2 = $t0 << 1；$t2 = $t2 | $t1. 第一步操作后的结果为：

$t0 = 0x 0000 A5A5, $t1 = 0x 0000 5A5A

$t2 = (0000 0000 0000 0000 1010 0101 1010 0101) << 1

= (0000 0000 0000 0001 0100 1011 0100 1010) = 0x 0001 4B4A

再将$t2寄存器内的值与$t1按位或：

$$\$t2 = \begin{array}{l} 0000\ 0000\ 0000\ 0001\ 0100\ 1011\ 0100\ 1010 \\ 0000\ 0000\ 0000\ 0000\ 0101\ 1010\ 0101\ 1010 \end{array} | \\ = 0000\ 0000\ 0000\ 0001\ 0101\ 1011\ 0101\ 1010 = 0x\ 0001\ 5B5A$$

② 对于*b.*情况中的指令含义为：$t2 = $t0 >> 1；$t2 = $t2 & (0x00F0). 第一步操作后的结果为：

$t0 = 0x 0000 A5A5, $t1 = 0x 0000 5A5A

$t2 = (0000 0000 0000 0000 1010 0101 1010 0101) >> 1

= (0000 0000 0000 0000 0101 0010 1101 0010) = 0x 0000 52D2

再将$t2寄存器内的值与(0x00F0)按位与：

$$\$t2 = \begin{array}{l} 0000\ 0000\ 0000\ 0000\ 0101\ 0010\ 1101\ 0010 \\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1111\ 0000 \end{array} \& \\ = 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1101\ 0000 = 0x\ 0000\ 00D0$$

(5) $\$t0 = 0\text{x A5A5 0000}, \$t1 = 0\text{x A5A5 0000}$

① 对于a.情况中的指令含义为：$\$t2 = \$t0 << 1$ ；$\$t2 = \$t2 \mid \$t1$. 第一步操作后的结果为：

$\$t0 = 0\text{x A5A5 0000}$, $\$t1 = 0\text{x A5A5 0000}$

$\$t2 = (1010\ 0101\ 1010\ 0101\ 0000\ 0000\ 0000\ 0000) << 1$

$= (0100\ 1011\ 0100\ 1010\ 0000\ 0000\ 0000\ 0000) = 0\text{x 4B4A 0000}$

再将$\$t2$寄存器内的值与$\$t1$按位或：

$$\$t2 = \begin{array}{l} 0100\ 1011\ 0100\ 1010\ 0000\ 0000\ 0000\ 0000 \\ 1010\ 0101\ 1010\ 0101\ 0000\ 0000\ 0000\ 0000 \end{array} \mid$$

$$= 1110\ 1111\ 1110\ 1111\ 0000\ 0000\ 0000\ 0000 = 0\text{x EFEF 0000}$$

② 对于b.情况中的指令含义为：$\$t2 = \$t0 >> 1$ ；$\$t2 = \$t2$ & (0x00F0). 第一步操作后的结果为：

$\$t0 = 0\text{x A5A5 0000}$, $\$t1 = 0\text{x A5A5 0000}$

$\$t2 = (1010\ 0101\ 1010\ 0101\ 0000\ 0000\ 0000\ 0000) >> 1$

$= (0101\ 0010\ 1101\ 0010\ 1000\ 0000\ 0000\ 0000) = 0\text{x 52D2 8000}$

再将$\$t2$寄存器内的值与(0x00F0)按位与：

$$\$t2 = \begin{array}{l} 0101\ 0010\ 1101\ 0010\ 1000\ 0000\ 0000\ 0000 \\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1111\ 0000 \end{array} \&$$

$$= 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000 = 0\text{x 0000 0000}$$

(6) $\$t0 = 0\text{x A5A5 FFFF}, \$t1 = 0\text{x A5A5 FFFF}$

① 对于a.情况中的指令含义为：$\$t2 = \$t0 << 1$ ；$\$t2 = \$t2 \mid \$t1$. 第一步操作后的结果为：

$\$t0 = 0\text{x A5A5 FFFF}$, $\$t1 = 0\text{x A5A5 FFFF}$

$\$t2 = (1010\ 0101\ 1010\ 0101\ 1111\ 1111\ 1111\ 1111) << 1$

$= (0100\ 1011\ 0100\ 1011\ 1111\ 1111\ 1111\ 1110) = 0\text{x 4B4B FFFE}$

再将$\$t2$寄存器内的值与$\$t1$按位或：

$$\$t2 = \begin{array}{l} 0100\ 1011\ 0100\ 1011\ 1111\ 1111\ 1111\ 1110 \\ 1010\ 0101\ 1010\ 0101\ 1111\ 1111\ 1111\ 1111 \end{array} \mid$$

$$= 1110\ 1111\ 1110\ 1111\ 1111\ 1111\ 1111\ 1111 = 0\text{x EFEF FFFF}$$

② 对于b.情况中的指令含义为：$\$t2 = \$t0 >> 1$ ；$\$t2 = \$t2$ & (0x00F0). 第一步操作后的结果为：

$\$t0 = 0\text{x A5A5 FFFF}$, $\$t1 = 0\text{x A5A5 FFFF}$

$\$t2 = (1010\ 0101\ 1010\ 0101\ 1111\ 1111\ 1111\ 1111) >> 1$

$= (0101\ 0010\ 1101\ 0010\ 1111\ 1111\ 1111\ 1111) = 0\text{x 52D2 FFFF}$

再将$\$t2$寄存器内的值与(0x00F0)按位与：

$$\$t2 = \begin{array}{l} 0101\ 0010\ 1101\ 0010\ 1111\ 1111\ 1111\ 1111 \\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1111\ 0000 \end{array} \&$$

$$= 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1111\ 0000 = 0\text{x 0000 00F0}$$

2.21.

假设栈和静态数据段都是空的并且栈指针和全局指针依次指向地址0x 7FFF FFFC 和0x 1000 8000。调用习惯如图2 − 11,函数的输入使用寄存器 $a0,返回值使用 $v0。假定页函数仅可以使用保留寄存器。试回答下面问题。

a.

```
main(){
   leaf_function(1);
}
int leaf_function(int f){
   int result;
      result = f + 1;
      if(f > 5)   return result;
   leaf_function(result);
}
```

b.

```
int  my_global = 100
main() {
   int x = 10;   int y = 20;
   int z;
   z = my_function(int x, int y);
}
int my_function(int x, int y){
   return  x − y;
}
```

2. 21. 1 [5] < 2.8>画出每次函数调用后栈和静态数据段的内容。

解：(1) 假定程序需要借助帧指针实现，以下为程序执行过程中栈中的情况：

$sp = 0x 7FFF FFFC →

$ra
$fp
main()局部变量以及预留空间
$a0
$ra
$fp
函数局部变量以及预留空间
函数递归

$ra
$fp
main()局部变量以及预留空间
$a1
$a0
$fp
函数局部变量以及预留空间
函数递归

数据段通常是指用来存放程序中已初始化的全局变量的一块内存区域。数据段属于静态内存分配。对于*a*程序静态数据段没有用到，而对于*b*程序，全局指针=0x1000 8000，静态数据段内存有全局变量my_global以及它的值 100。

以下为 MIPS 汇编的冒泡排序，实现输入数据的降序排列：

完整代码与注释附于其后。

其中 Syscall根据$v0 的值不同而执行不同的效果：

1：输出$a0 中的值；

4：输出给定字符串；

5：将控制台的数据读入$v0；

11：打印给定字符；

30：计算代码执行时间

程序运行示例如下：

可见四个数的排序用时较少，在毫秒级体现不出来