HW2

周恩帅 zes18119713525@mail.ustc.edu.cn

2.10.1-3

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.	$1010\ 1110\ 0000\ 1011\ 0000\ 0000\ 0000\ 0100_2$
b.	$1000\ 1101\ 0000\ 1000\ 0000\ 0000\ 0100\ 0000_2$

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。

- R-Type

6	5	5	5	5	6
op	rs	rt	rd	shamt	funct

- I-Type

6	5	5	16
op	rs	rt	Constant or Address
101011	10000	01011	0000 0000 0000 0100
sw	$16($s0)	$11($t3)	4
100011	01000	01000	0000 0000 0100 0000
lw	$8($t0)	$8($t0)	64

sw $t3, 4($s0)　　lw $t0, 64($t0)

2.10.4-6

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

2.10.4 [5] <2.4, 2.5>求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、rs 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。

- R-Type

add $t0 $t0 $zero 0x01004020

op	rs	rt	rd	shamt	funct
0x0	0x8	0x0	0x8	00000	0x20

- I-Type

lw $t1, 4($s3) 0x8e690004

op	rs	rt	Constant or Address
0x23	0x13	0x9	0x4

MIPS指令类型

- R-Type

6	5	5	5	5	6
op	rs	rt	rd	shamt	funct

- I-Type　addi、lui、lw、sw、beq(字地址偏移)

6	5	5	16
op	rs	rt	Constant or Address

- J-Type　J、Jal

6	26
op	Address

26位形式地址左移2位，与PC的高4位拼接

2.13.1-3

a.	$t0 = 0x55555555, $t1 = 0x12345678
b.	$t0 = 0xBEADFEED, $t1 = 0xDEADFADE

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
```

Basic	
lui $1, 0xffffffff	1: andi $t2, $t2, -1
ori $1, $1, 0x0000ffff	
and $10, $10, $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
```

andi $10, $10, 0x0000ffef	1: andi $t2, $t2, 0xFFEF

2.13.1
a. 0x57755778
b. 0xFEFFFEDE
2.13.2
a. 0x00005550(0x55555550)
b. 0x0000EED0(0xEADFEED0)
2.13.3
a. 0x0000AAAA
b. 0x0000BFCD

SRA	算数右移 shift right arithmetic， 高位补符号位
SRL	逻辑右移 shift right logical，高位补0
SLL	逻辑左移 shift left logical， 地位补0
ADDI/ADDIU	符号扩展/零扩展（16位立即数）
ANDI/ORI	零扩展（16位立即数）
LW/SW/BEQ	符号扩展（16位立即数）

- 汇编器行为：（不截断）
Lui填$1高16位，ori填低16位
构造$1 = -1, 再进行and
得出$t2=0x55555550
- 或者截断，认为-1=0xFFFF
Addi单指令编码立即数为0xFFFF
零扩展后得 0x0000FFFF
得出$t2=0x00005550
- 0xFFEF 必然是零扩展

2.13.4-6

2.13.4
a. 0x00015B5A
b. 0x000000D0
2.13.5
a. 0xEFEF0000
b. 0x00000000
2.13.6
a. 0xEFEFFFFF
b. 0x000000F0

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

2.13.4 [5] <2.6>假设 $t0 =0x0000A5A5， $t1 =00005A5A。求执行上表中的指令后最终寄存器 $t2 中的值。

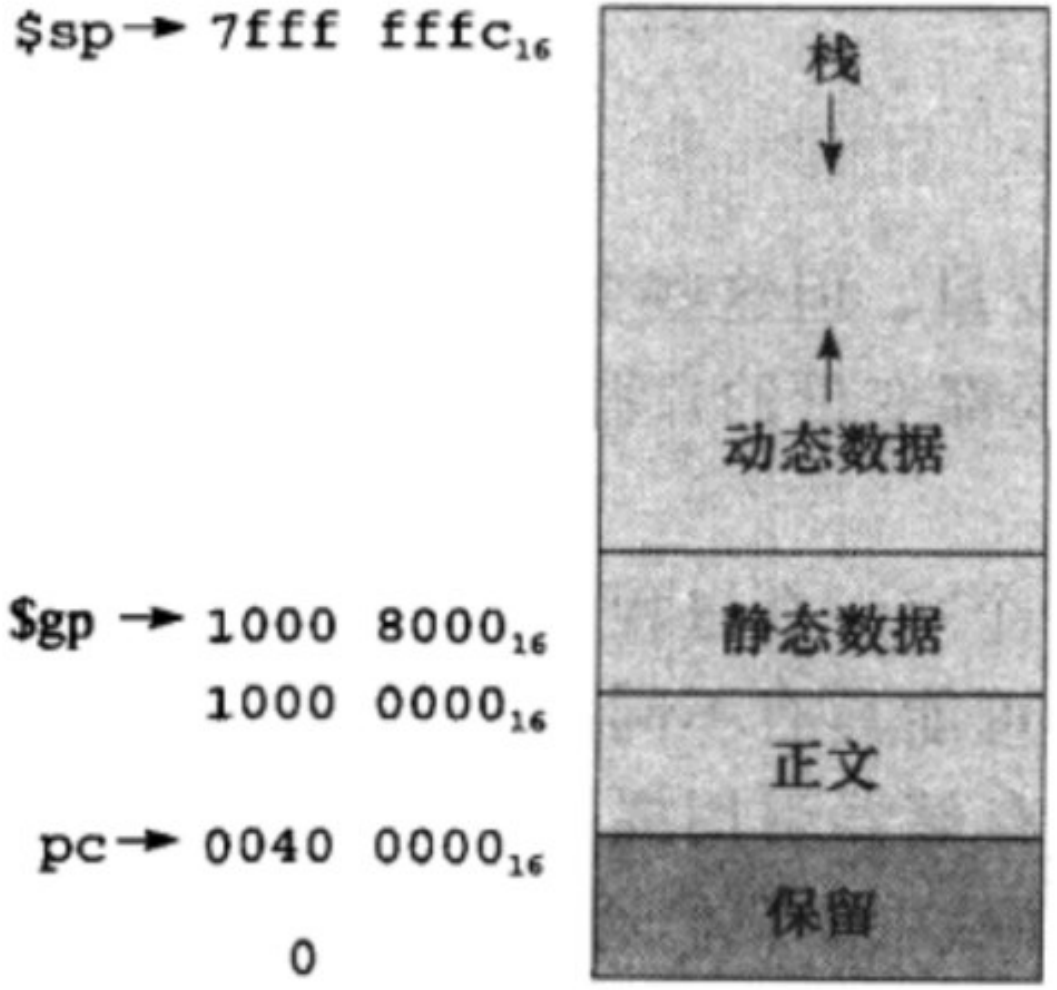
2.13.5 [5] <2.6>假设 $t0 =0xA5A50000， $t1 = A5A50000。求执行上表中的指令后最终寄存器 $t2 中的值。

2.13.6 [5] <2.6>假设 $t0 =0xA5A5FFFF， $t1 = A5A5FFFF。求执行上表中的指令后最终寄存器 $t2 中的值。

b	$t2
srl $t2, $t0, 1	0000 0000 0000 0000 1010 0101 1010 0101 >> 1 = 0000 0000 0000 0000 0101 0010 1101 0010
andi $t2, $t2, $t1	0x000052D2 & 0x000000F0 = 0x000000D0

进程内存布局

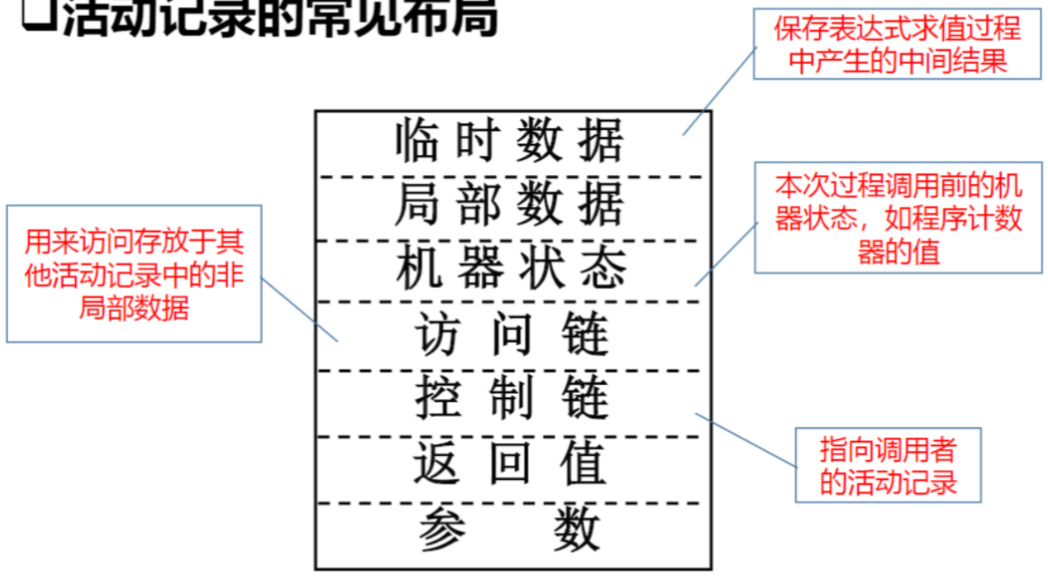
- 栈 stack，高地址向低地址延伸
- 堆 heap，动态数据区，动态内存分配，malloc，new等。与栈增长方向相反
- 代码段，存放程序指令。
- 静态数据段，存储常量和其他静态变量。如static修饰的变量，全局变量。编译时就已经分配空间。通过全局指针$gp+正负16位偏移 访问。



栈结构

- 栈指针$sp，指向栈顶
- 过程帧，活动记录，过程所保存的寄存器和局部变量的片段
- 帧指针$fp, 指向过程帧的第一个字
- $s0-$s7, 保留寄存器，被调者保存
- $t0-$t9, 临时寄存器，被调者不必保存

❑活动记录的常见布局



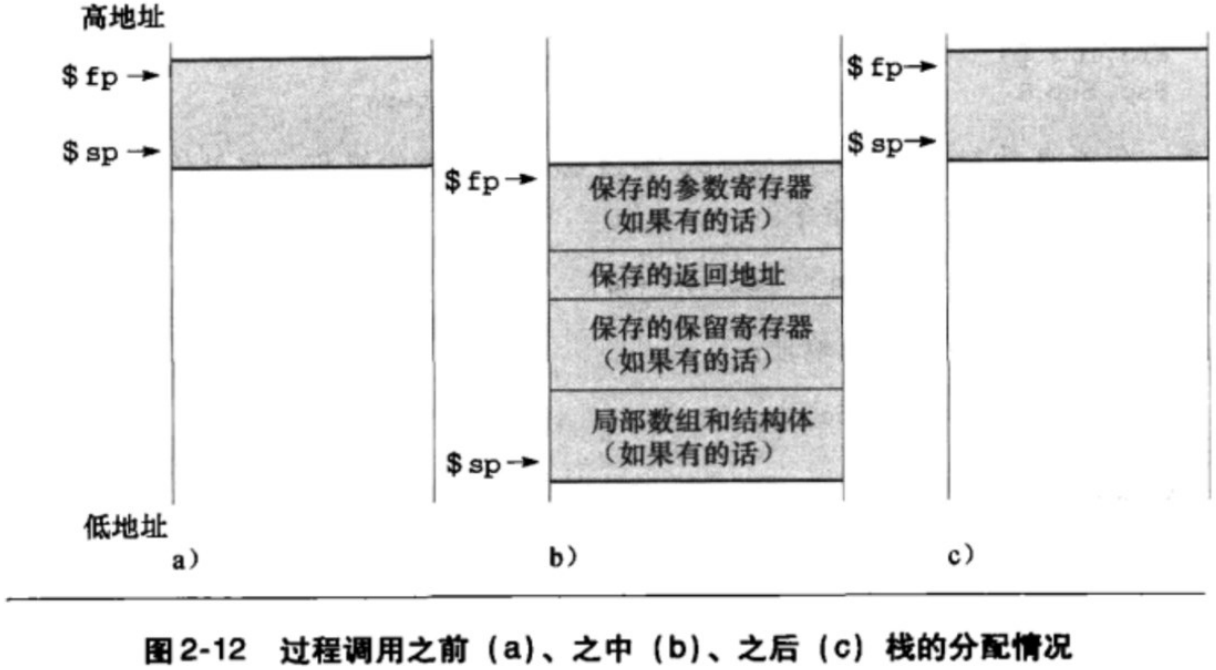


图2-12 过程调用之前（a）、之中（b）、之后（c）栈的分配情况

2.21.1

a.

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

进入main	地址	内容
	0x7fff fffc	？？？
	-4	返回地址 $ra->
$fp, $sp->	-8	保存的$fp
	…	
$gp->	0x1000 8000	？？？

保留	不保留
保存寄存器：$s0 ~ $s7	临时寄存器：$t0 ~ $t9
栈指针寄存器：$sp	参数寄存器：$a0 ~ $a3
返回地址寄存器：$ra	返回值寄存器：$v0 ~ $v1
栈指针以上的栈	栈指针以下的栈

图 2-11　过程调用时保留和不保留的内容

2.21.1

a.

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

进入leaf_function(1)	地址	内容
	0x7fff fffc	？ ？ ？
	-4	返回地址 $ra->
	-8	保存的$fp
	-12	返回地址 $ra ->main
$fp->	-16	保存的$fp
$sp->	-20	result
	…	
$gp->	0x1000 8000	？ ？ ？

2.21.1

a.

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

进入leaf_function(2)	地址	内容
	0x7fff fffc	？？？
	-4	返回地址 $ra->
	-8	保存的$fp
	-12	返回地址 $ra ->main
leaf_function(1)	-16	保存的$fp
	-20	result
	-24	保存的参数 $a0 = f？
	-28	返回地址$ra ->leaf(1)
$fp->	-32	保存的$fp
$sp->	-36	result
	…	
$gp->	0x1000 8000	？？？

2.21.1

b.

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

进入main	地址	内容
	0x7fff fffc	？？？
	-4	返回地址 $ra
$fp->	-8	保存的$fp
	-12	x
	-16	y
$sp ->	-20	z
	…	
$gp->	0x1000 8000	my_global

2.21.1

b.

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

进入my_function	地址	内容
	0x7fff fffc	？？？
	-4	返回地址 $ra
	-8	保存的$fp
	-12	x
	-16	y
	-20	z
	-24	返回地址 $ra ->main
$fp->	-28	保存的$fp
$sp->		临时值 x-y
	…	
$gp->	0x1000 8000	my_global