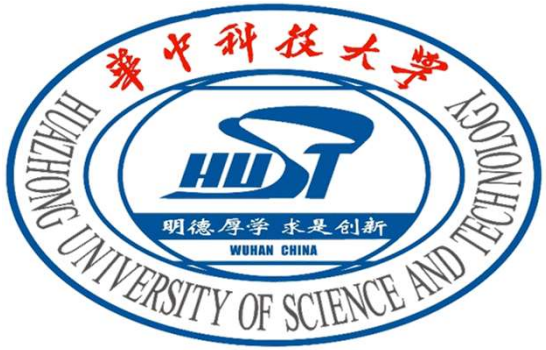
微机原理与接口技术

子程序原理

华中科技大学　左冬红



主、子程序

程序是完成某个任务的指令序列

调用、被调用关系

```
int sum(int a,int b);
int diff(int a,int b);
float exp(int base,int power);
```

```
int main()
 {    int a=2;
      int b=4;
      float c;
      c=exp(a,b);
      return 0;
}
```

主程序

```
float exp(int a,int b)
{     int base[2];
      base[0]=sum(a,b);
      base[1]=diff(a,b);
      return (base[0]*base[0])+(base[1]*base[1]);
}
```

嵌套调用子程序

子程序

主程序

```
int sum(int a,int b)
{     return a+b;
}
```

```
int diff(int a,int b)
{     return a-b;
}
```

子程序

主、子程序执行过程

```
int main()
 {    int a=2;
      int b=4;
      float c;
      c=exp(a,b);
      return 0;
}
```

```
float exp(int a,int b)
{     int base[2];
      base[0]=sum(a,b);
      base[1]=diff(a,b);
      return (base[0]*base[0])+(base[1]*base[1]);
}
```

```
int sum(int a,int b)
{     return a+b;
}
```

```
int diff(int a,int b)
{     return a-b;
}
```

程序执行过程中需跳转入子程序或从子程序跳转回

程序执行过程中需传参数给子程序或从子程序传回结果

计算机硬件如何实现这些功能？

主、子程序执行过程

```
int main()
{    int a=2;
     int b=4;
     float c;
     c=exp(a,b);
     return 0;
}
```

```
float exp(int a,int b)
{    int base[2];
     base[0]=sum(a,b);
     base[1]=diff(a,b);
     return (base[0]*base[0])+(base[1]*base[1]);
}
```

```
int sum(int a,int b)
{    return a+b;
}
```

```
int diff(int a,int b)
{    return a-b;
}
```

转入子程序

子程序名称与调用名称一致

子程序名称即标号

无条件跳转指令

从子程序转回

返回到下一条语句

如何知道下一条语句地址？

转入子程序前保存PC的值

转入子程序前，PC自动指向下一条语句

转入子程序后，PC指向子程序下一条语句

主、子程序转返指令

	指令格式	指令功能
转入子程序	jal Label	保存PC的值到$ra,并跳转到Label
从子程序转回	jr $ra	将$ra的值赋给PC
一般格式	jr $Rs	间接跳转

主、子程序参数传递

$2~$3	**$v0~$v1**	**函数调用返回值(values for results and expression evaluation)**
$4-$7	$a0-$a3	函数调用参数(arguments)

C编译器按照参数定义顺序依序使用寄存器

$8-$15	**$t0-$t7**	**临时寄存器(temporary)**
$16-$23	$s0-$s7	存储寄存器(saved)，C语言中定义的变量可以保存在这些寄存器中。同时这些寄存器也可以保存存储单元的起始地址（基地址）
$24-$25	**$t8-$t9**	**临时寄存器(temporary)**

若寄存器数目不够，使用存储器传递参数

存储器只能使用地址寻址，参数传递仅在调用和返回时发生，存储地址无需事先分配，因此动态分配存储空间

主、子程序MIPS汇编指令实现示例

主程序

```
float exp(int a,int b)
{     int base[2];
      base[0]=sum(a,b);
      base[1]=diff(a,b);
      return (base[0]*base[0])+(base[1]*base[1]);
}
```

```
jal sum
sw $v0,0($s0)
jal diff
sw $v0,4($s0)
```

子程序

```
int sum(int a,int b)
{     return a+b;
}
```

```
add $t0,$a0,$a1
add $v0,$t0,$0
jr $ra
```

```
int diff(int a,int b)
{     return a-b;
}
```

```
sub $t0,$a0,$a1
add $v0,$t0,$0
jr $ra
```

主、子程序MIPS汇编指令实现示例

```
float exp(int a,int b)
{     int base[2];
      base[0]=sum(a,b);
      base[1]=diff(a,b);
      return (base[0]*base[0])+(base[1]*base[1]);
}
```

```
jal sum
sw $v0,0($s0)
jal diff
sw $v0,4($s0)
```

可行吗？

```
int main()
 {    int a=2;
      int b=4;
      float c;
      c=exp(a,b);
      return 0;
}
```

$ra

$a0=2,$a1=4

```
float exp(int a,int b)
{     int base[2];
      base[0]=sum(a-1,b+1);
      base[1]=diff(a-2,b+3);
      return (a*base[0]*base[0])+(b*base[1]*base[1]);
}
```

$ra

$ra

$a0=1,$a1=5

$a0=0,$a1=7

子程序嵌套调用时，$a0,$a1值发生变化

子程序嵌套调用时，$ra值发生变化

主、子程序公用寄存器

子程序中先保存，然后才能使用　　保存在栈中

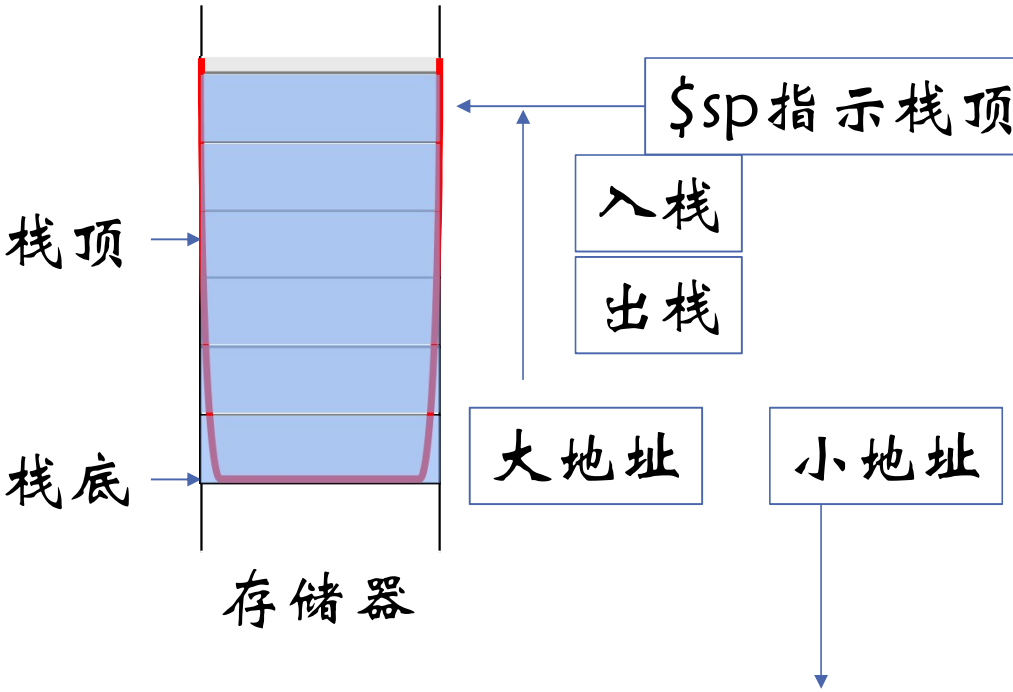
$16-$23	**$s0-$s7**	存储寄存器(saved)，C语言中定义的变量可以保存在这些寄存器中。同时这些寄存器也可以保存存储单元的起始地址（基地址）

$2~$3	**$v0~$v1**	**函数调用返回值(values for results and expression evaluation)**
$4-$7	$a0-$a3	函数调用参数(arguments)

$30	**$fp**	**帧指针(frame pointer)**
$31	$ra	返回地址(return address)

主、子程序动态存储空间——栈

栈：存储器中开辟的一片数据存储区，一端固定，另一端活动，且只允许数据从活动端进出，先进后出



栈中数据地址基于$sp、$fp寻址

微处理器有专门栈操作指令，则寄存器$sp隐含在栈操作指令中

微处理器无专门栈操作指令，如MIPS微处理器，寄存器$sp需显式指出，栈中数据访问指令仍为lw,sw

MIPS C编译器子程序嵌套调用栈管理规则

主、子程序栈空间连续，使用前分配

分配栈空间　　addi $sp,$sp,-32

子程序执行过程中$sp不变

子程序栈空间

$fp指示当前子程序可操作栈顶

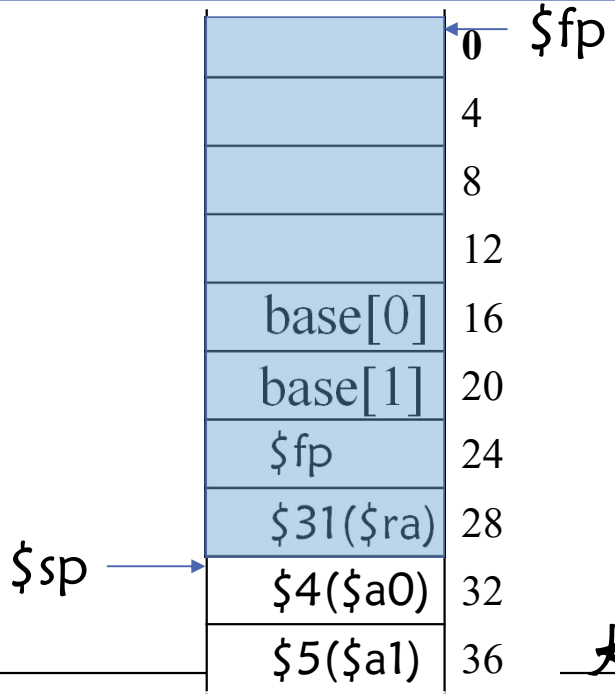
$ra,$fp等采用入栈操作保存

其余寄存器保存基于$fp寻址

主程序栈空间

子程序嵌套调用示例

```
float exp(int a,int b)
{     int base[2];
      base[0]=sum(a-1,b+1);
      base[1]=diff(a-2,b+3);
      return (a*base[0]*base[0])+(b*base[1]*base[1]);
}
```




```
exp: addi $sp,$sp,-32
     sw   $31,28($sp)
     sw   $fp,24($sp)
     add  $fp,$sp,$0
     sw   $4,32($fp)
     sw   $5,36($fp)
     lw   $2,32($fp)
     addi $3,$2,-1
     lw   $2,36($fp)
     addi $2,$2,1
     add  $5,$2,$0
     add  $4,$3,$0
     jal sum
     sw $2,16($fp)
     lw $2,32($fp)
     addi $3,$2,-2
     lw $2,36($fp)
     addi $2,$2,3
     add  $5,$2,$0
     add  $4,$3,$0
     jal  diff
     sw   $2,20($fp)
     lw   $3,16($fp)
     lw   $2,32($fp)
     mul  $3,$3,$2
     lw   $2,16($fp)
     mul  $3,$3,$2
     lw   $4,20($fp)
     lw   $2,36($fp)
     mul  $4,$4,$2
     lw   $2,20($fp)
     mul  $2,$4,$2
     addu $2,$3,$2
     add $sp,$fp,$0
     lw   $31,28($sp)
     lw   $fp,24($sp)
     addiu $sp,$sp,32
     jr   $31
```

小结

http://reliant.colab.duke.edu/c2mips/

- 主、子程序互转指令
- 参数传递
  - 寄存器
- 栈
  - 栈顶$sp，数据先进后出，入、出必须配对
- 公用寄存器栈保存
- GCC编译器栈管理规则、寄存器使用规则