COMPUTER ORGANIZATION (IS F242)

LECT 26: MIPS ARCHITECTURE

Leaf Procedure Example

- C code:

```
int leaf_example (int g, int h,
                              int i, int j)
{ int f;
  f = (g + h) - (i + j);
  return f;
}
```

  - Arguments g, …, j in \$a0, …, \$a3
  - f in \$s0 (hence, need to save \$s0 on stack)
  - Result in \$v0

Leaf Procedure Example

MIPS code:

```
leaf_example:
addi $sp, $sp, -4
sw   $s0, 0($sp)          Save $s0 on stack

add  $t0, $a0, $a1
add  $t1, $a2, $a3        Procedure body
sub  $s0, $t0, $t1

add  $v0, $s0, $zero      Result

lw   $s0, 0($sp)
addi $sp, $sp, 4          Restore $s0

jr   $ra                  Return
```

Non-Leaf Procedures

- Procedures that call other procedures
- For nested call, caller needs to save on the stack:
  - Its return address
  - Any arguments and temporaries needed after the call
- Restore from the stack after the call

Non-Leaf Procedure Example

- C code:

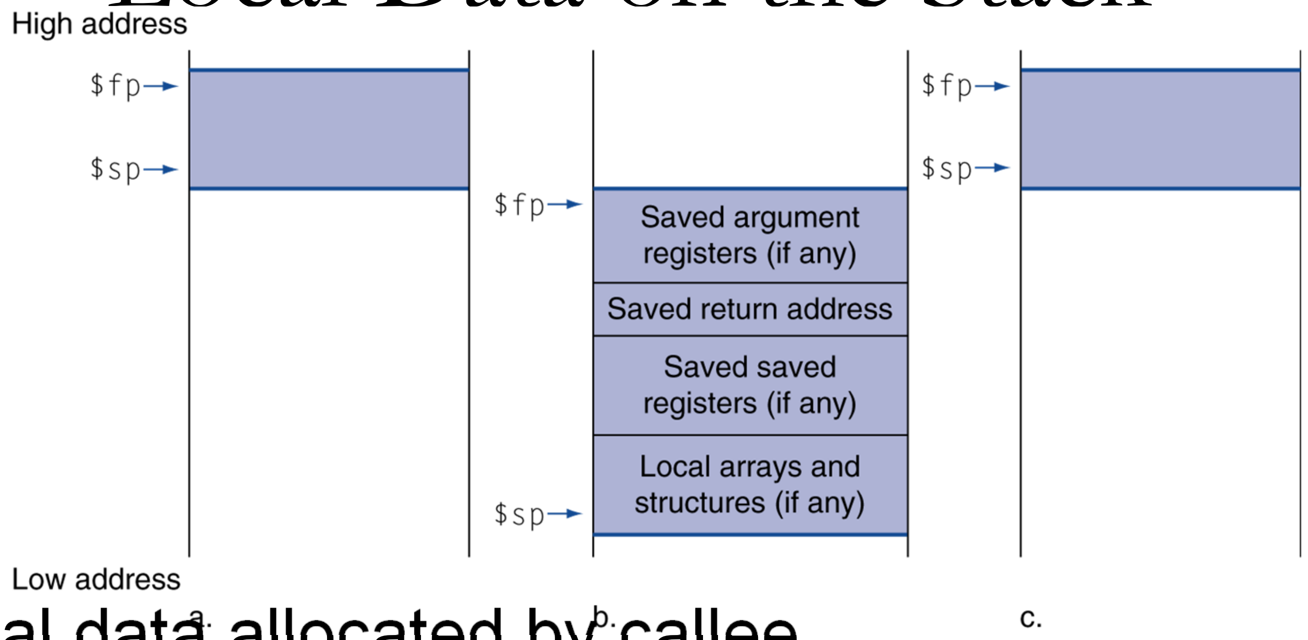
```
int fact (int n)
{
  if (n < 1) return 1;
  else return n * fact(n - 1);
}
```

  - Argument n in $a0
  - Result in $v0

Non-Leaf Procedure Example

```
fact:
    addi $sp, $sp, -8     # adjust stack for 2 items
    sw   $ra, 4($sp)      # save return address
    sw   $a0, 0($sp)      # save argument
    slti $t0, $a0, 1      # test for n < 1
    beq  $t0, $zero, L1
    addi $v0, $zero, 1    # if so, result is 1
    addi $sp, $sp, 8      #   pop 2 items from stack
    jr   $ra              #   and return
L1: addi $a0, $a0, -1     # else decrement n
    jal  fact             # recursive call
    lw   $a0, 0($sp)      # restore original n
    lw   $ra, 4($sp)      #   and return address
    addi $sp, $sp, 8      # pop 2 items from stack
    mul  $v0, $a0, $v0    # multiply to get result
    jr   $ra              # and return
```

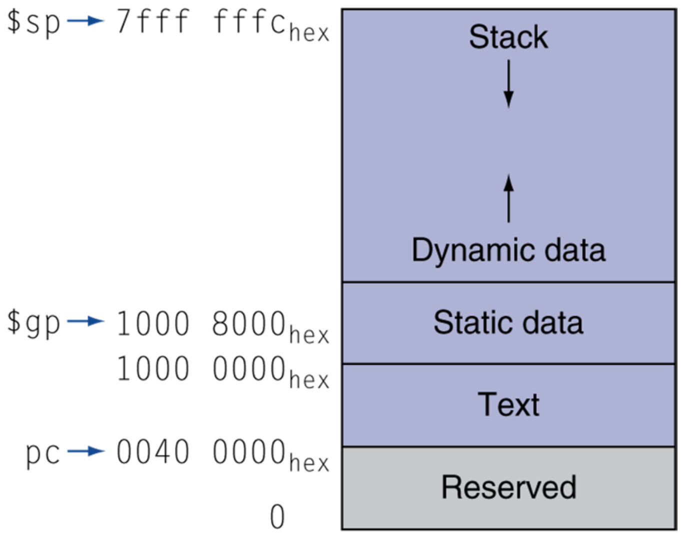
Local Data on the Stack


High address
$fp
$sp
$fp
$sp
$fp
$sp
Saved argument registers (if any)
Saved return address
Saved saved registers (if any)
Local arrays and structures (if any)
Low address
a.
b.
c.

- Local data allocated by callee
  - e.g., C automatic variables
- Procedure frame (activation record)
  - Used by some compilers to manage stack storage
- Frame pointer
  - points to the 1st word of the frame of a procedure
  - Value denoting the location of the saved registers and local variables for a given procedure

Memory Layout

- Text: program code
- Static data: global variables
  - e.g., static variables in C, constant arrays and strings
  - $gp initialized to address allowing ±offsets into this segment
- Dynamic data: heap
  - E.g., malloc in C, new in Java
- Stack: automatic storage



String Copy Example

- C code (naïve):
  - Null-terminated string

```
void strcpy (char x[], char y[])
{ int i;
  i = 0;
  while ((x[i]=y[i])!='\0')
    i += 1;
}
```

  - Addresses of x, y in $a0, $a1
  - i in $s0