

ECE 3140/CS 3420 Computer Organization Spring 2009

Procedures

ECE3140/CS3420



Cornell University

Announcements

- Homework 1 is done
 - Congratulations on very few CMS problems!
- Project 1
 - Due Tue., Feb 10 at 10:00pm
 - Don't wait until the last minute to post to CMS!
 - Problems in convert.s have been fixed
- You will need an ECE account on amdpool to do the projects for this class
 - <https://accounts.ece.cornell.edu/>
 - Please do this now, *not* Feb 10!

ECE3140/CS3420



Cornell University

2

Hennessy and Patterson

- Read Chapter 1
 - 1.1-1.9
- Read Chapter 2
 - 2.1 through 2.14,
 - Skim B.1,B.2, B.10
- Read
 - Appendix B.6
 - 3.1 through 3.2
 - MIPS Calling Convention Document (website)
- Read (for Tuesday)
 - Appendix B.1-B.5
 - Notes on Programming in C (website)

ECE3140/CS3420



Cornell University

3

Procedure Calls & Returns

```
main:
:
la $4, s
jal NumSpaces

:

ori $4,$2,0
jal puts

:

li $4, 0
jal exit
```

Call

Return

Call

Return

Call

```
Numspaces:
li $2, 0
lbu $8, 0($4)

:

jr $31
```

```
puts:
lbu $8, 0($4)

:

jr $31
```

```
exit:
:

syscall
```

Procedure Calls

- Jump to start
- Parameters
- Register Management

Procedure Returns

- Return from end
- Return values
- Register Management

MIPS support

- Very little HW support
- Mostly Convention
- Caller/Callee Contract

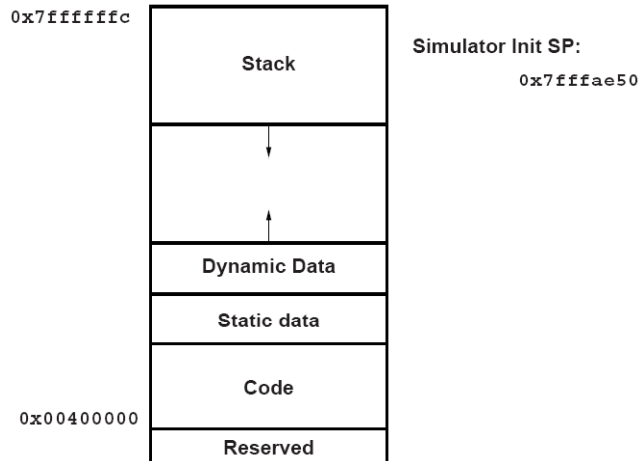
ECE3140/CS3420



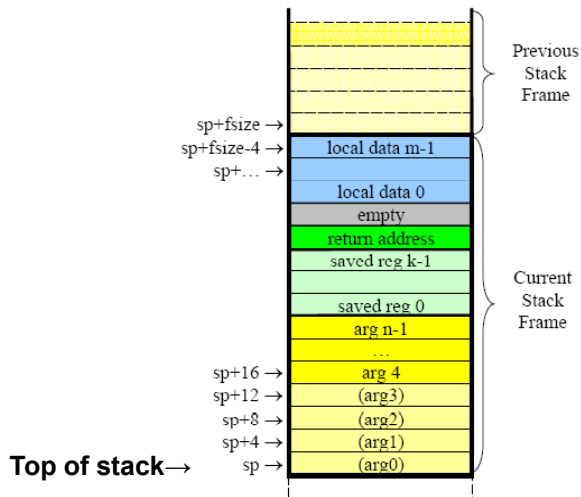
Cornell University

4

Memory Layout



Our Stack Frames



Simple Leaf Routine

```
int g( int x, int y ) {  
    return (x + y);  
}
```

```
g:    add    $v0,$a0,$a1 # result is sum of args  
      jr    $ra # return
```

- Stack Frame Design
 - Local Storage: NO - Everything fits in registers
 - Pad: NO - Not needed
 - Return Address: NO - Leaf Routine
 - Saved Registers NO - No Saved Regs used
 - Argument Block: NO - Leaf Routine
- Stack Size = 32 words (128 bytes)

ECE3140/CS3420



Cornell University

7

Leaf Routine with Local Data

```
int g( int x, int y ) {  
    int a[32];  
    ... (calculate using x, y, a);  
    return a[0];  
}
```

- Stack Frame Design
 - Local Storage: YES - Local data cannot fit in registers
 - Pad: NO - Not needed
 - Return Address: NO - Leaf Routine
 - Saved Registers NO - Not needed
 - Argument Block: NO - Leaf Routine
- Stack Size = 32 words (128 bytes)

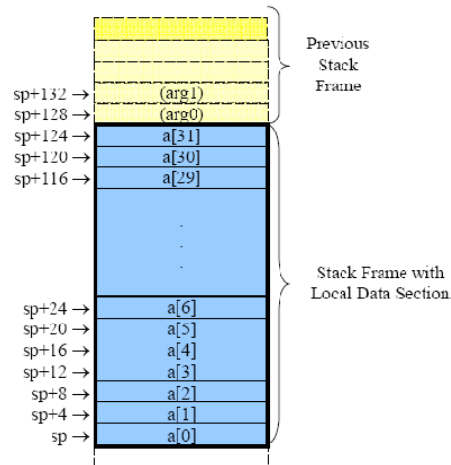
ECE3140/CS3420



Cornell University

8

Leaf Routine with Local Data



ECE3140/CS3420



Cornell University

9

Prologue/Epilogue

- Prologue sets up stack frame for this call
- Epilogue undoes stack frame before returning to caller
- Keeping stack manipulation instructions together simplifies code
- Everything between the Prologue and the Epilogue is the Body

ECE3140/CS3420



Cornell University

10

Leaf Routine with Local Data

```
g:      # start of prologue
        addiu $sp,$sp,(-128) # push stack frame
        # end of prologue

        . . .                # calculate using $a0, $a1 and a
                               # array a is stored at addresses
                               # 0($sp) to 124($sp)

        lw   $v0,0($sp)      # result is a[0]

        # start of epilogue
        addiu $sp,$sp,128    # pop stack frame
        # end of epilogue

        jr   $ra             # return
```

ECE3140/CS3420



Cornell University

11

Leaf Routine w/ Saved Regs

```
int g( int x, int y) {
    int a,b,c; // use $s1 for a, $s3 for b, $s4 for c
    ...
    // calculation using x, y, a, b, c;

    return a+b+c;
}
```

• Stack Frame Design

- Local Storage: NO - Everything fits in registers
- Pad: YES - Space for 1 word
- Return Address: NO - Leaf Routine
- Saved Registers YES - Space for 3 words
- Argument Block: NO - Leaf Routine
- Stack Size = 4 words (16 bytes)

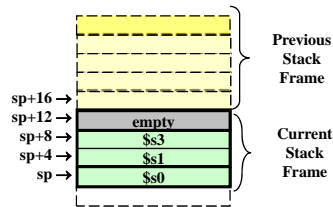
ECE3140/CS3420



Cornell University

12

Leaf Routine w/ Saved Regs



Leaf Routine w/ Saved Regs

```

g:
    # start of prologue
    addiu  $sp,$sp,-16    # push stack frame
    sw     $s1, 0($sp)    # save value of $s1
    sw     $s3, 4($sp)    # save value of $s3
    sw     $s4, 8($sp)    # save value of $s4
    # end of prologue

    # start of body
    ...
    addu   $v0,$s1,$s3
    addu   $v0,$v0,$s4
    # end of body

    #start of epilogue
    lw     $s4, 8($sp)    # restore value of $s4
    lw     $s3, 4($sp)    # restore value of $s3
    lw     $s1, 0($sp)    # restore value of $s1
    addiu  $sp,$sp,16    # pop stack frame

    jr     $ra
    
```



Non-Leaf Routine

```
int g( int x, in y) {  
    ...  
    // contains calls to f(x), h(x,y), s(x,2*x,y), t(x+y,x,y,x-y)  
    ...  
    return t(x+y,x,y,x-y);  
}
```

- Stack Frame Design
 - Local Storage: NO - Everything fits in registers
 - Pad: YES - Space for 1 word
 - Return Address: YES - Non-Leaf Routine
 - Saved Registers: NO - Not needed
 - Argument Block: YES - Minimum of 4 words
- Stack Size = 6 words (24 bytes)

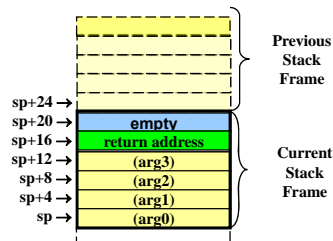
ECE3140/CS3420



Cornell University

15

Non-Leaf Routine



ECE3140/CS3420



Cornell University

16

Non-Leaf Routine

```
g:  # start of prologue
    addiu $sp,$sp,-24  # push stack frame
    sw    $31,16($sp)  # save return address
    # end of prologue

    # start of body
    jal   f
    ...
    jal   h
    ...
    jal   s
    ...
    jal   t
    # end of body

    #start of epilogue
    lw    $ra, 16($sp)  # restore value of $s1
    addiu $sp,$sp,0     # pop stack frame

    jr    $ra
```

ECE3140/CS3420



Cornell University

17

Exercise

- Construct a stack frame for the following:

```
int ece314(int a,b,c; char *s, int *p) {
    int x[5];
    int aa, bb, cc;
    :
    aa = f(a,b,c);
    :
    bb = g(a,b,c,s,p);
    :
    cc = h(s,p);
    :
    return aa + bb + cc;
}
```

**Assume that we
need 5 saved
registers (\$s0-\$s4)**

ECE3140/CS3420



Cornell University

NumSpaces()

```
int NumSpaces(char *s) {  
    int count;  
    if (!(*s)) return 0;  
    count = NumSpaces(s+1);  
    if (!(*s == ' ')) {  
        count++;  
    }  
    return(count);  
}
```

• Stack Frame Design

- Local Storage: NO - Everything fits in registers
- Pad: NO - Not Needed
- Return Address: YES - Non-Leaf Routine
- Saved Registers: YES - Space for 1 words
- Argument Block: YES - Non-Leaf Routine
- Stack Size = 6 words (24 bytes)

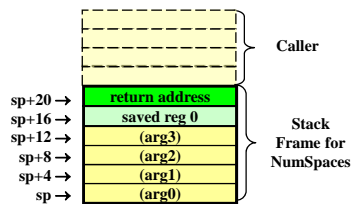
ECE3140/CS3420



Cornell University

22

NumSpaces()



ECE3140/CS3420



Cornell University

23

NumSpaces()

NumSpaces:

```
# Prologue
    addiu $sp, $sp, -24    # Push Stack Frame
    sw $ra, 20($sp)      # Save return address
    sw $s0, 16($sp)      # Save $s0

# Body
    li $v0, 0            # $v0: why?
    lbu $s0, 0($a0)      # $s0 = *s
    blez $s0, $done      # goto $done
    addiu $a0, $a0, 1    # s++
    jal NumSpaces
    li $t0, 32           # $t1 = ``
    bne $s0, $t0, $done
    addiu $v0, $v0, 1    # $v0 = count++
    ...
```

ECE3140/CS3420



Cornell University

24

NumSpaces()

```
...
# Epilogue
$done: lw $s0, 16($sp)    # Restore $s0
       lw $ra, 20($sp)   # Restore return address
       addiu $sp, $sp, 24 # Pop Stack Frame

       jr $ra           # Return
```

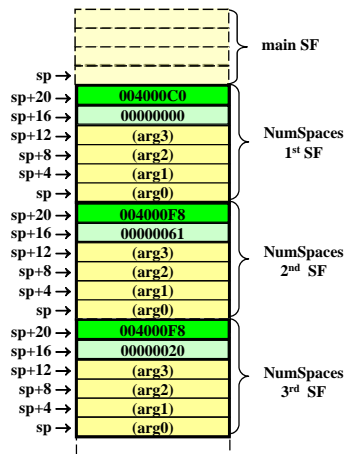
ECE3140/CS3420



Cornell University

25

Recursive Execution



```

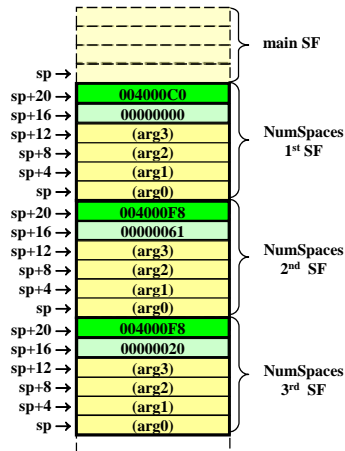
la $a0, s          # main program
jal NumSpaces
    addiu $sp, $sp, -24
    sw $ra, 20($sp)
    sw $s0, 16($sp)
    ...
    jal NumSpaces
        addiu $sp, $sp, -24
        sw $ra, 20($sp)
        sw $s0, 16($sp)
        ...
        jal NumSpaces
            addiu $sp, $sp, -24
            sw $ra, 20($sp)
            sw $s0, 16($sp)
            ...
            jal NumSpaces
                ...
                ...
    
```

ECE3140/CS3420



Cornell University

Recursive Execution (cont)



```

...
    lw $s0, 16($sp)
    lw $ra, 20($sp)
    addiu $sp, $sp, 24
    jr $ra
    ...
    lw $s0, 16($sp)
    lw $ra, 20($sp)
    addiu $sp, $sp, 24
    jr $ra
    ...
    lw $s0, 16($sp)
    lw $ra, 20($sp)
    addiu $sp, $sp, 24
    jr $ra
    li $a0, 0          # main program
    
```

ECE3140/CS3420



Cornell University

Procedures, Recap

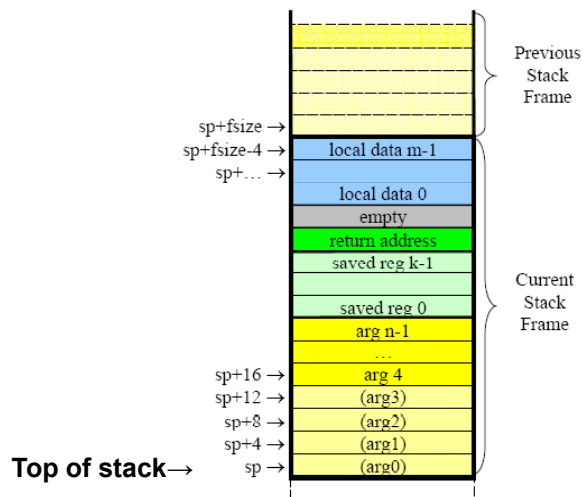
- A *stack frame* holds information about each procedure call
 - Each call pushes a new stack frame onto the stack
 - Each return pops the stack frame off the stack
- This information *may* include local data, return address, saved variables, and arguments
- Registers used by procedure calls/returns are \$a0,\$a1,\$a2,\$a3,\$ra, and \$sp
- Stack frames must always be double word aligned

ECE3140/CS3420



Cornell University

Our Stack Frames



ECE3140/CS3420



Cornell University

Local Data Section

- You *must* include a Local Data Section in the stack frame for a procedure if:
 - The procedure requires more storage than can fit in available registers
- The Local Data Section *must* always be
 - Double word aligned
 - Have a size that is a multiple of 8

ECE3140/CS3420



Cornell University

Return Address Slot

- You *must* include a Return Address Slot in the stack frame for a procedure if:
 - The procedure calls another procedure (i.e., is a non-leaf procedure)
- The return address *must* be copied from \$31 (\$ra) to the stack in the prologue
- The return address *must* be copied from the stack to \$31 (\$ra) in the epilogue
- A pad *may* be required above the return address slot to keep the Local Data doubleword aligned.

ECE3140/CS3420



Cornell University

Saved Registers Section

- You *must* include a Saved Registers Section in the stack frame for a procedure if:
 - The procedure body uses (i.e., changes the value of) registers \$s0 to \$s7
- The values of saved registers *must* be copied to the stack in the prologue
- The values of saved registers *must* be copied from the stack in the epilogue

ECE3140/CS3420



Cornell University

Arguments Section

- You *must* include an Arguments Section in the stack frame for a procedure if:
 - The procedure calls any procedure with 1 or more arguments
- The arguments section *must* be large enough for *all* parameters of *any* called procedure
- Arguments 1-4 are passed in registers \$a0-\$a3; Arguments 5 and up are copied into the argument section
- The procedure *must not* put anything in the first four argument slots; the called procedures *may* copy the values of their first four parameters into the first four argument slots.

ECE3140/CS3420



Cornell University