CS2318

20/06/2016

Function Call (using stack)

Chapter 6 Britton

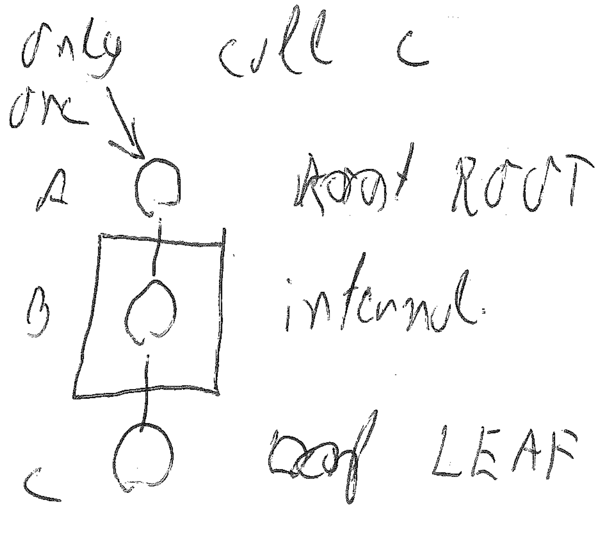
chapter 2 patterson

B:

(function)

Jal C

JR ~~$~~ RA



2 Consider an internal node

A function that is both a caller and a callee

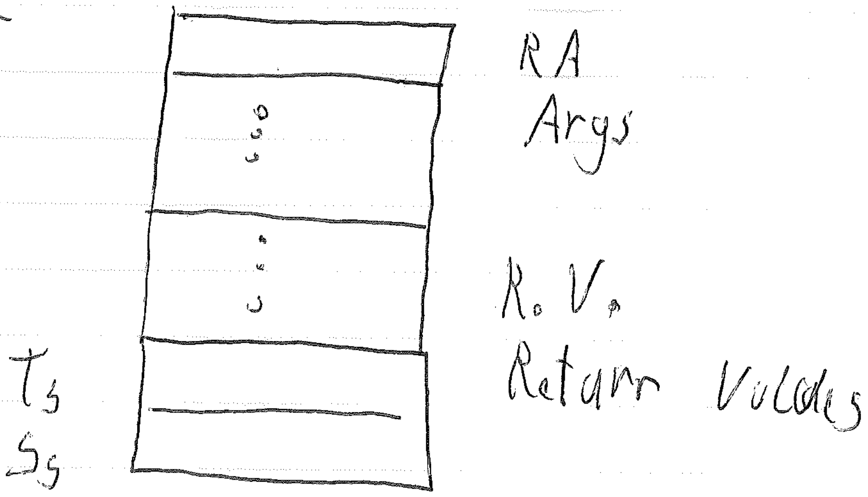
caller	callee

B:	C:

```
[ do something
JAL C
[ do somethin
ret
```

```
[ do somethin
ret
```

From for B calling C



caller

B:

push \$RA (Into frame on stack)

Push ARGS

push Needed T_s (C do not "care" about B's T's) (Like Local vars)

JAL C

Pop \$RA

Pop Return values

pop T_s

Return

callee

C:

pop ARGS

push Needed S_s

(can use T_s, S_s)

[illegible]

[compute
  Get value

push Return Value

pop S_s

return

4 Int_Sqrt (n)

```
  :
              check me
  While((i x i) <= n ) {
  i = i+1;
  }
  ;
```

Main Gets an int n

checks if it is prime

```
      While (i <= Int_Sqrt (n) )
    { check rem
      if 0 break;
    }
```

5

```
Main:                          Assuming
        Get n <-> $s0          Root

        ~~Fn~~ :     push $RA
               :     push $s0

        Find sqrt(n)
               :         pop $RA
               :         pop R.V. <-> $s1
   $t0 ———————
        while (i < sqrt(n))
          {
          check Rem
          if (Rem == 0) break, [illegible] prime
           i = i+1;
        }
          prime

   not prime
```

Q6 SQRT(n) Assuming Leaf

```
  :    POP $S0
  :
$t0 ___
  While((ixi) < n) {
    i = i+1;
  }
  :
  :
  ___
  JR $RA
           push result
```

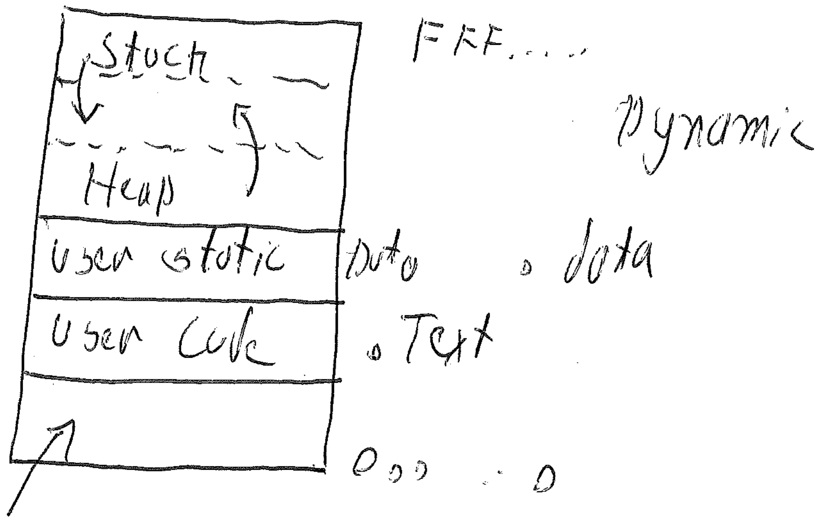
Push R_s Pop R_d

Using $SP as the stack pointer

Ignoring underflow / overflow

7 ~~8~~

Typical Memory Organization



"OS" code   System Code

down/Up in addressed

Stack can grow up / Down

SP can point to current data

Can point to next available

space

8

Assum stack grows down
$sp point to next availble
space

push $R_s   pop $Rd

```
sw $R_s, 0($sp)       subi $sp, $sp, 4
addi $sp, $sp, 4      lw $Rd, 0($sp)
```

push $t0, $t1, $t2

```
sw $t0, 0($sp)
sw $t1, 4($sp)
sw $t2, 8($sp)
addi $sp, $sp, 12 ← advance
by "frame" size
```

9

Assum stack grows down

$sp points to current data

push $Rs	pop $Rd
add	Load
then	then
stor	subtract.

while ($\underbrace{(\overset{\$t0}{i} \times i)}_{\$t1}$ < n) {

i = i + 1;

}

```
       li $t0, 1

       mul $t1, $t0, $t0

while: bge $t1, $s0, endwhile

       addi $t0, $t0, 1
       mul $t1, $t0, $t0
       b while
endwhile
```