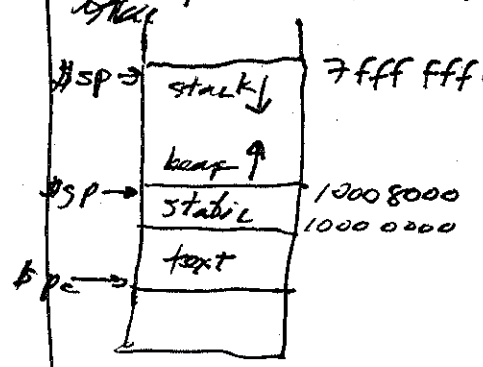
Stack

~~[illegible]~~ procedures

interrupts/exceptions 6.7.2, 7.7

Memory Alloc fig 6.31, "green card"



Stack grows down from 0x7fff

points to ~~[illegible]~~ lowest used

procedure ~~[illegible]~~ calls section 6.4.6

no args, no return value

args, return value

local & global variables

fig 6.23 (modified)

```
main() {
  simple();
}

void simple(){
  return;
}
```

```
0x 00   main: jal simple
                nop

0x1000  simple: jr $ra
                nop
```

what goes into $ra in jal?

What is the line ~~after~~ at simple+4 is jr $0?

$\underbrace{000000}_{op}\ 00000\ 00000\ \underbrace{01000}_{0x8\ funct}$

simple args, return value

```
int y;                  0x1000  addi $a0, $0, 3
                        1004    addi $a1, $0, 8;
y = plus(3,8);          1008    jal plus
                        100c    nop
                        1010    add $s0, $v0, $0

int plus(int a,b){      2000 plus: add $v0, $a0, $a1
                                   jr $ra
  return a+b;                      nop
}
```

// $s0 = y)

could we get rid of the nop after jal? no

after jr? no

what is in $ra at 0x2008 1010

could we swap 1004 & 1008? yes

could we swap 100c & 1010? no

more than 4 args:

1st 4 go in $a0-3

rest go on stack.

what if I need $t0? use it. Caller responsibility

what, if I need $s0?, $ra?, $a0? } must save on stack.

$v0? expected.

what, if I need extra space? local vars on stack

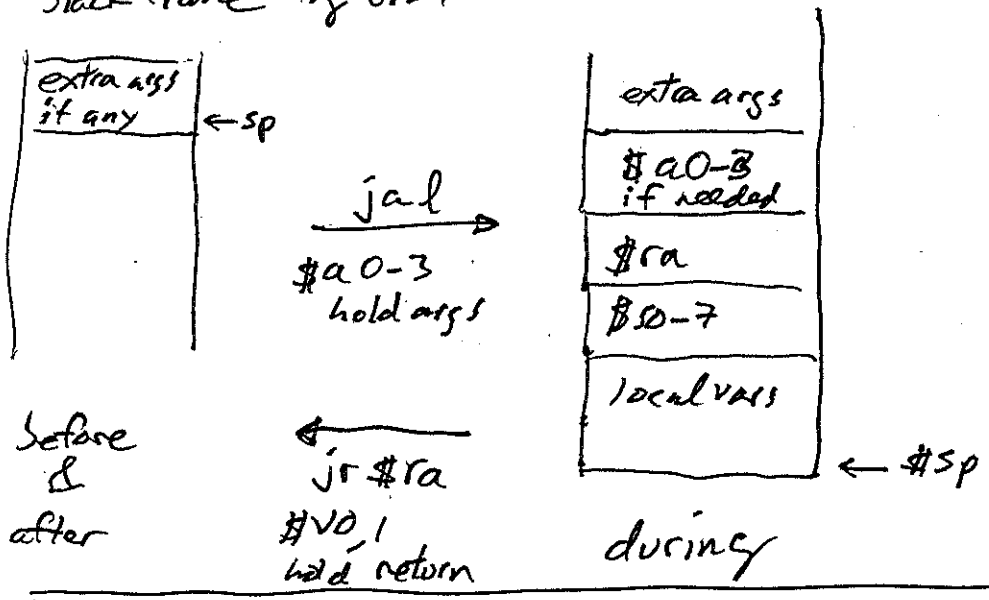
When would I need to save $ra?

If procedure calls another.

Extra space

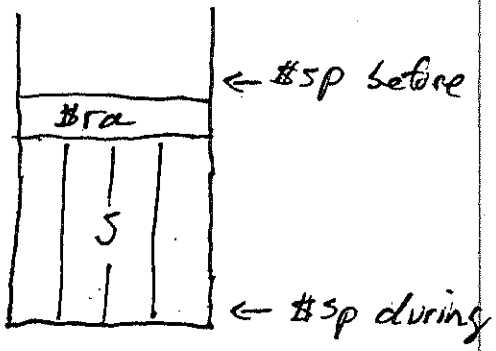
```
int function(){
  int a,b;        ← will go in $s0, $s1
  int c[10];      ← won't fit

                    addi $sp, $sp, -10
c[0] = a;           sw $s0, 0($sp)
c[9] = b;           sw $s1, 36($sp)
}
```

Stack frame fig 6.27



EX:

```
        7fff 0040
int input(){        3c
  char s[20];       38
  char              34
  gets(s);          30
                    2c
}                   28
```



what if input is

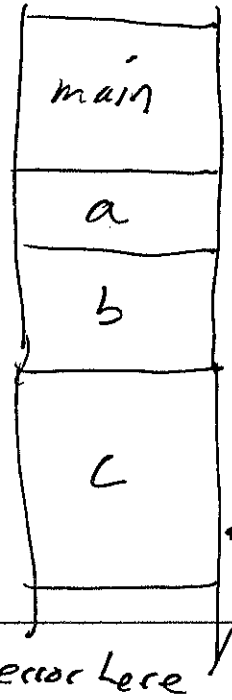
0x [ ] 7fff028 [ ]

more chars hackers! 20 chars

~~the seq~~

```
main(){
  a( )
}

a( ){
  b( )
}

b( ){
  c( )
}
```

```
c( ){
}
```



stack frames

args & ra are on the stack.

~~[illegible]~~ is easy to trace.

error here

globals/static/extern

int x,y,z; ⇐ these go in "static data" segment.

main()

volatile int a; ⇐ tells compiler to load/store every time it gets used.