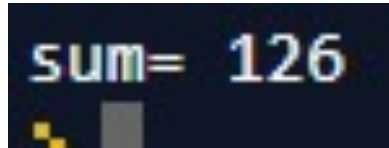


Lab7

Add numbers between 2 integer

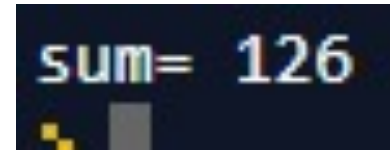
```
int main(){  
    int sum=0;  
    int a=10;  
    int b=18;  
  
    while(a<=b){  
        sum=sum+a;  
        a=a+1;  
    }  
    printf("sum= %d\n",sum);  
    return 0;  
}
```

A terminal window with a dark background showing the output of the program: "sum= 126". The text is in a light blue/cyan color. There are some small yellow and grey artifacts at the bottom left of the terminal window.

sum= 126

Add numbers between 2 integer with function call

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(){
5      int sum = 0;
6      int a = 10;
7      int b=18;
8
9      addfunc(&sum,&a,b);
10     printf("sum = %d",sum);
11
12     return 0;
13 }
14
15 void addfunc(int *sum,int *a,int b){
16     while (*a<=b){
17         *sum=*sum+*a;
18         *a+=1;
19     }
20 }
```

A terminal window with a dark background showing the output of the program: "sum= 126". The text is in a light blue/cyan monospace font. There are some small yellow and grey artifacts on the left side of the terminal window.

Fill Array with integers

```
1  #include <stdio.h>
2
3  int main3(void) {
4      int list[25] = {};
5      int t=0;
6
7      while (t<25){
8          list[t]=t;
9          t=t+1;
10     }
11
12
13     return 0;
14 }
15
```

Fill Array with integers with function call

```
1
2  #include <stdio.h>
3
4  void fillArray(int size,int list[]);
5
6  int main(void) {
7      int list[25] = {};
8      int t=0;
9
10     fillArray(25,list);
11
12
13     return 0;
14 }
15
16 void fillArray(int size,int list[]){
17     int t=0;
18     while (t<size){
19         list[t]=t;
20         t=t+1;
21     }
22 }
23
24
25 }
```

Stack

- Stack is a LIFO queue.
- A stack pointer ($\$sp$) contains the address of the most recently allocated address (*top of the stack*).
- Stack has two operations: “push” and “pop”.
- Stacks, in MIPS, grow from higher to lower addresses.
 - “push” decrement $\$sp$.
 - “pop” increment $\$sp$.

Example 1/2

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

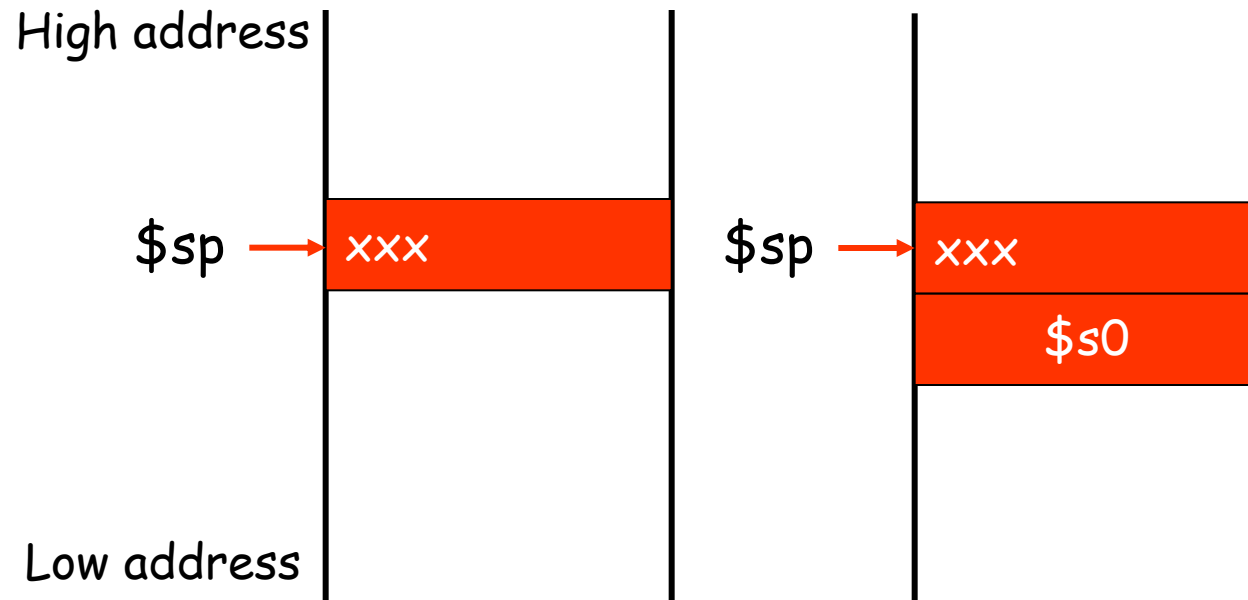
• g, h, i, j → \$a0, \$a1, \$a2, \$a3 and f → \$s0

leaf_example:

```
addi $sp, $sp, -4    # make room for one item
sw   $s0, 0($sp)     # save register $s0
...
```

• Content of \$s0 is saved on stack since the callee is going to use it.

Stack Activity



Example 2/2

...

\$s0 saved

add \$t0, \$a0, \$a1 # \$t0 = g+h

add \$t1, \$a2, \$a3 # \$t1 = i+j

sub \$s0, \$t0, \$t1 # \$s0 = \$t0 - \$t1

add \$v0, \$s0, \$zero # \$v0 = \$s0 + 0 (returns f)

...

- before returning, restore the old values of the registers for the caller

...

lw \$s0, 0(\$sp) # restore register \$s0

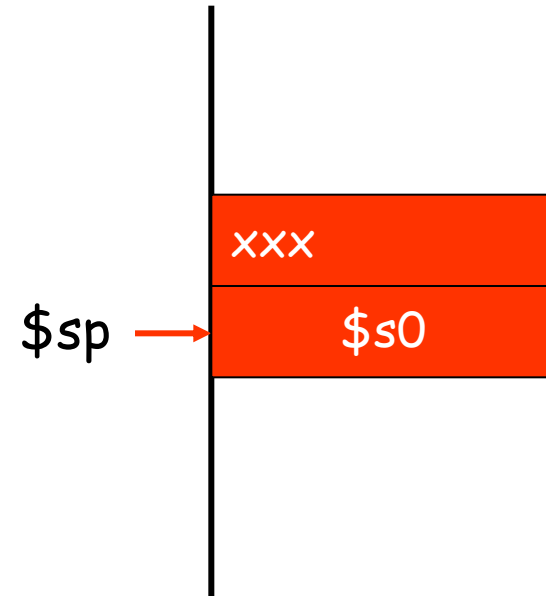
add \$sp, \$sp, 4 # delete 1 item

...

- return the control to the caller

jr \$ra # jump back to the caller

Stack Activity



Fill Array with integers with function call+Print

```
1  #include <stdio.h>
2
3  void fillArray(int size,int list[]);
4  void printArray(int size,int list[]);
5
6  int main(void) {
7      int list[25] = {};
8      int t=0;
9
10     fillArray(25,list);
11     printArray(25,list);
12
13     return 0;
14 }
15
16 void fillArray(int size,int list[]){
17     int t=0;
18     while (t<size){
19         list[t]=t;
20         t=t+1;
21     }
22 }
23
24 void printArray(int size,int list[]){
25     int t=0;
26     while (t<size){
27         printf("%d\n",list[t]);
28         t=t+1;
29     }
30 }
```

Fill Array with integers with function call+Print+Swap?

```
1
2  #include <stdio.h>
3
4  void fillArray(int size,int list[]);
5  void printArray(int size,int list[]);
6  void swap (int v[], int k);
7
8  int main(void) {
9      int list[25] = {};
10     int t=0;
11
12     fillArray(25,list);
13     swap(list,10);
14     printArray(25,list);
15
16     return 0;
17 }
18
19 ⊕ void fillArray(int size,int list[]){ ...
25 }
26
27 ⊕ void printArray(int size,int list[]){ ...
33 }
34
35 void swap (int v[], int k)
36 {
37     int temp;
38     temp = v[k];
39     v[k] = v[k+1];
40     v[k+1] = temp;
41 }
```