Lab7

Add numbers between 2 integer

```c
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;
}
```

```
sum= 126
```

Add numbers between 2 integer with function call

```c
#include <stdio.h>
#include <stdlib.h>

int main(){
    int sum = 0;
    int a = 10;
    int b=18;

    addfunc(&sum,&a,b);
    printf("sum = %d",sum);

  return 0;
}

void addfunc(int *sum,int *a,int b){
  while (*a<=b){
      *sum=*sum+*a;
      *a+=1;
    }
}
```

```
sum= 126
```

Fill Array with integers

```c
#include <stdio.h>

int main3(void) {
  int list[25] = {};
  int t=0;

  while (t<25){
    list[t]=t;
    t=t+1;
  }


  return 0;
}

```

Fill Array with integers with function call

```c

#include <stdio.h>

void fillArray(int size,int list[]);

int main(void) {
  int list[25] = {};
  int t=0;

  fillArray(25,list);


  return 0;
}

void fillArray(int size,int list[]){
  int t=0;
  while (t<size){
    list[t]=t;
    t=t+1;
  }
}


}
```

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

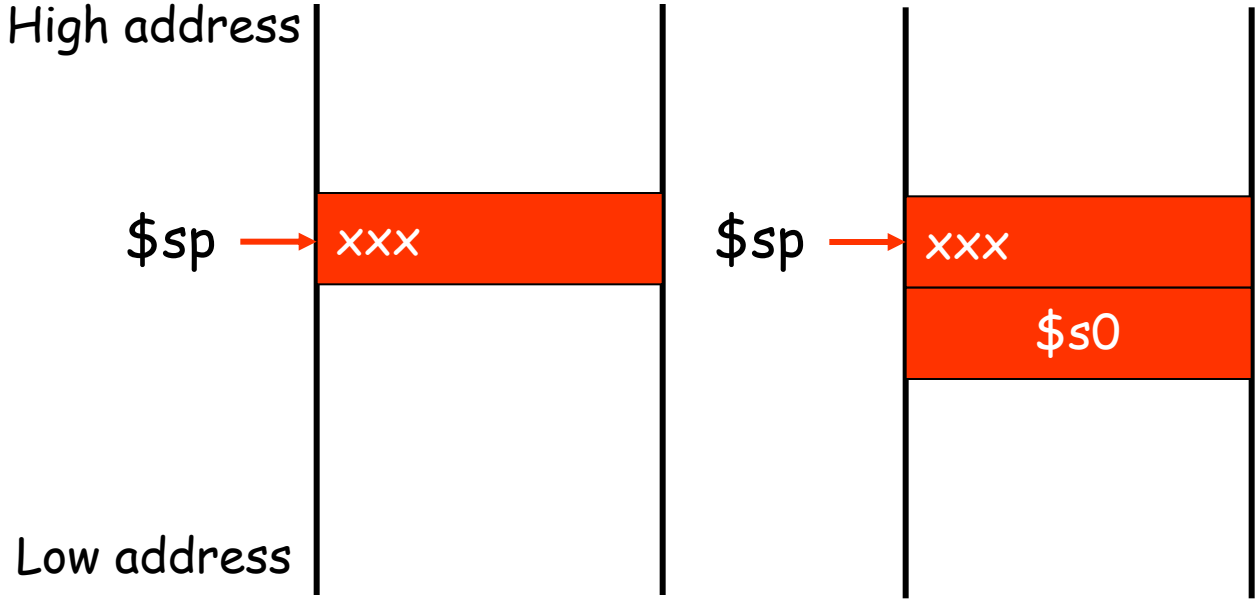
```
int leaf_example(int g, int h, int i, int j)
{
      int f;
      f = (g+h) - (i+j);
      return f;
}
```

- 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)
...
```

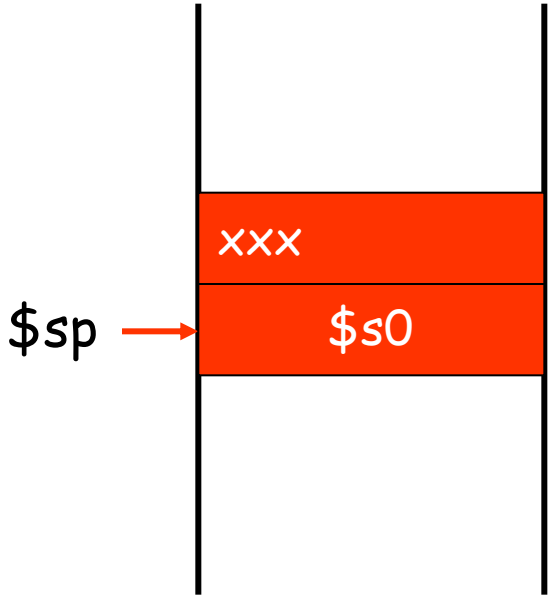
- before returning, restore the old values of the registers for the caller

```
...
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

```c
#include <stdio.h>

void fillArray(int size,int list[]);
void printArray(int size,int list[]);

int main(void) {
  int list[25] = {};
  int t=0;

  fillArray(25,list);
  printArray(25,list);

  return 0;
}

void fillArray(int size,int list[]){
  int t=0;
  while (t<size){
    list[t]=t;
    t=t+1;
  }
}

void printArray(int size,int list[]){
  int t=0;
  while (t<size){
    printf("%d\n",list[t]);
    t=t+1;
  }
}
```

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

```c
#include <stdio.h>

void fillArray(int size,int list[]);
void printArray(int size,int list[]);
void swap (int v[], int k);

int main(void) {
  int list[25] = {};
  int t=0;

  fillArray(25,list);
  swap(list,10);
  printArray(25,list);

  return 0;
}

void fillArray(int size,int list[]){ ···
}

void printArray(int size,int list[]){ ···
}

void swap (int v[], int k)
{
  int temp;
  temp = v[k];
  v[k] = v[k+1];
  v[k+1] = temp;
}
```