Funciones en C

Mg. Ing. Facundo S. Larosa

Informática I

Instituto Universitario Aeronáutico
Universidad de la Defensa Nacional (UNDEF)

¿Qué es una función?

1) Una función es un fragmento de código que posee un punto de entrada y un punto (o más de salida).

2) Puede recibir uno o más argumentos

3) Puede retornar un único valor

¿Por qué escribir funciones?

1) **Modularidad:** Dividir el código en módulos más pequeños facilita:
   1) Análisis
   2) Debugging
   3) Lectura y comprensión

2) **Reusabilidad:** Las funciones pueden agruparse en bibliotecas (*libraries*) las cuales pueden reusarse en futuras aplicaciones

3) **Eficiencia:** Las funciones pueden llamarse varias veces reduciendo el tamaño del código de programa.

Declaración de función

```
//Declaración de la función
[tipo devuelto] nombre_de_función (lista de parámetros)
{
   //Declaración de variables locales a la función

   //Código de la función

   //Retorno de valor (opcional)

}
```

Ejemplo: Código de función

```
#include <stdio.h>

int suma (int,int);

int main (void)
{
   int a=5,b=3,c;

   c=suma(a,b);

   printf("c=%d=%d+%d",c,a,b);
}
```

Prototipo

Argumentos

Llamada a función

```
int suma (int x, int y)
{
   int z;

   z=x+y;

   return z;
}
```

Parámetros formales

Código de la función

Retorno de valor

Ejemplo: Código de función

```
#include <stdio.h>

int suma (int,int);

int main (void)
{
   int a=5,b=3,c;

   c=suma(a,b);

   printf("c=%d=%d+%d",c,a,b);
}
```

Pasaje por valor

5

3

```
int suma (int x, int y)
{
   int z;

   z=x+y;

   return z;
}
```

A programar...

1) Escribir una función que recibe un número e indica si es primo (1) o no (0).

```
int esPrimo (int num)
```

2) Usar la función anterior para encontrar pares de números primos separados por dos unidades (ejemplo: 3 y 5, 5 y 7, etc.) desde 1 a 1000.

Tipos de variables

```
#include <stdio.h>

int suma (int,int);

int var;

int main (void)
{
   int a=5,b=3,c;

   c=suma(a,b);

   printf("c=%d=%d+%d",c,a,b);
}
```

Variable global

Variables locales de *main*

Parámetros formales de *suma*

```
int suma (int x, int y)
{
   int z;

   z=x+y;

   return z;
}
```

Variable local de *suma*

Tipos de variables

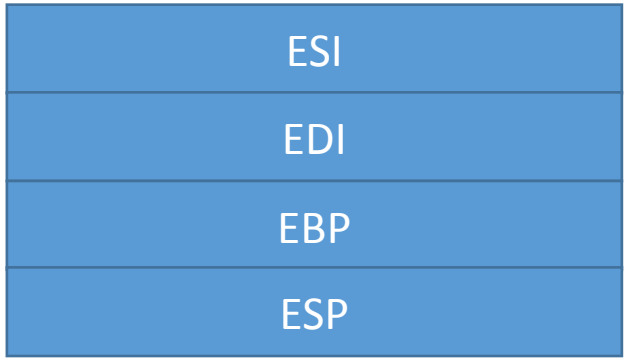
Variables	Valor inicial	Hábitat	Tiempo de vida	Visibilidad
Globales	Cero	Área estática	Todo el proceso	Todas las funciones *
Locales	Valor “aleatorio”	Área de pila asignada a la función	Vigencia de la función donde fueron creadas	Sólo en la función en que fueron declaradas
Parámetros formales	Argumento transferido			

* Salvo el caso de variables locales homónimas

Modelo del programador de Intel

Registros de propósito general

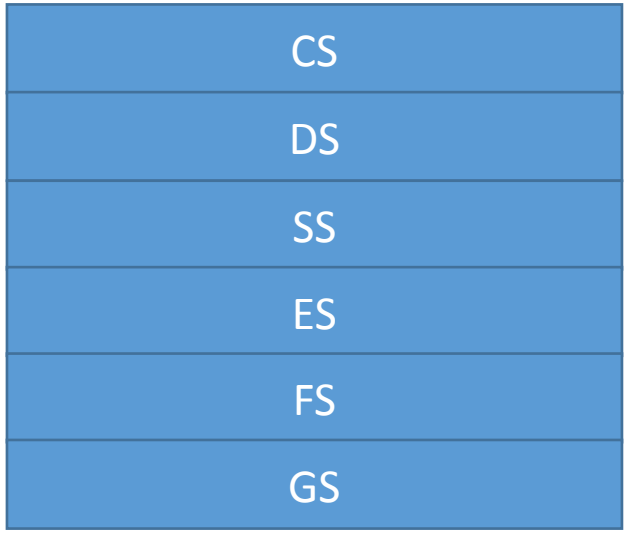
Registros de direccionamiento



Registro de control y estado

Puntero de programa

Registros de segmento



Internals de llamados a funciones

¿Qué es una pila?

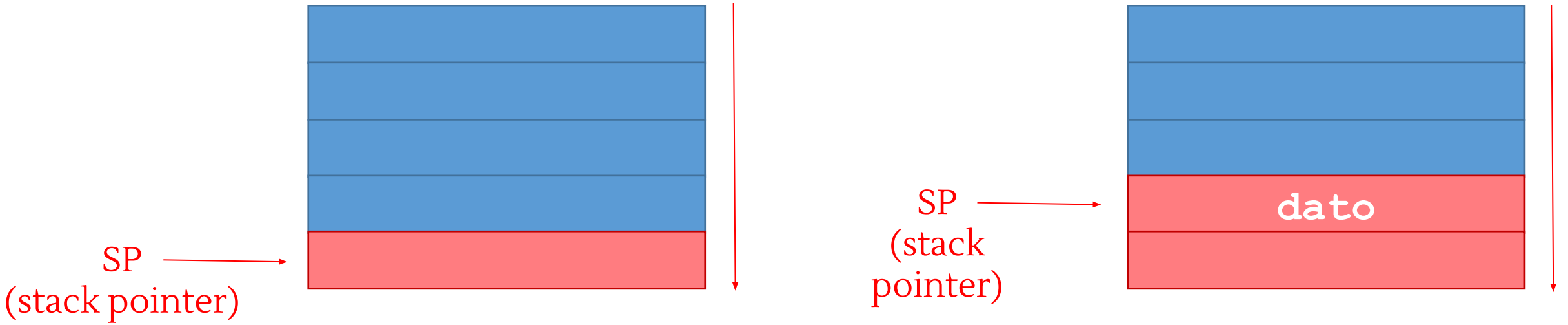
Es una estructura de datos de tipo LIFO (last in / first out), es decir, el último dato en entrar es el primero en salir.

Se opera sobre la pila ingresando o retirando datos, siendo la lectura destructiva.

Internals de llamados a funciones

Carga de datos en la pila (PUSH)

`PUSH dato`

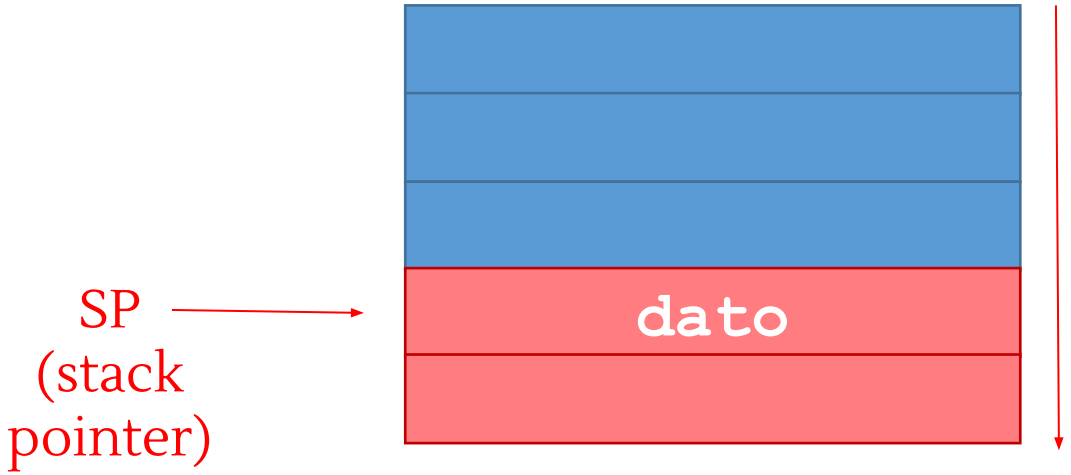


1. Se decrementa el SP (según tamaño del operando)
2. Se guarda el dato en la posición actual

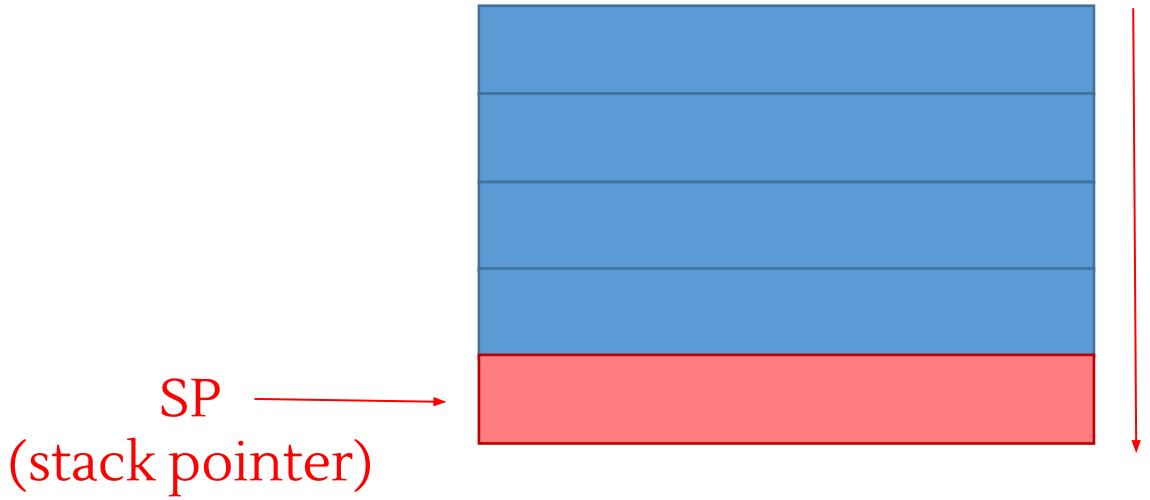
Internals de llamados a funciones

Lectura de datos de la pila (POP)

`POP dir`



1. Guarda el dato en la dirección (**`dir`**) indicada

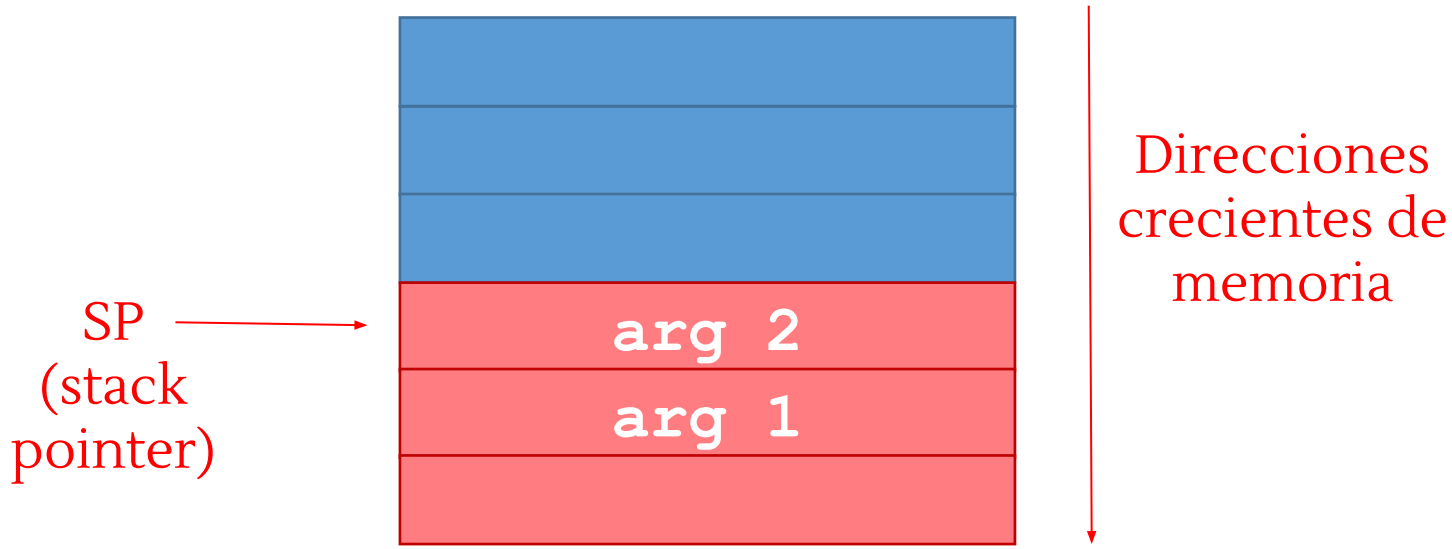


2. Se guarda el dato en la posición actual

Internals de llamados a funciones

Llamado a subrutina

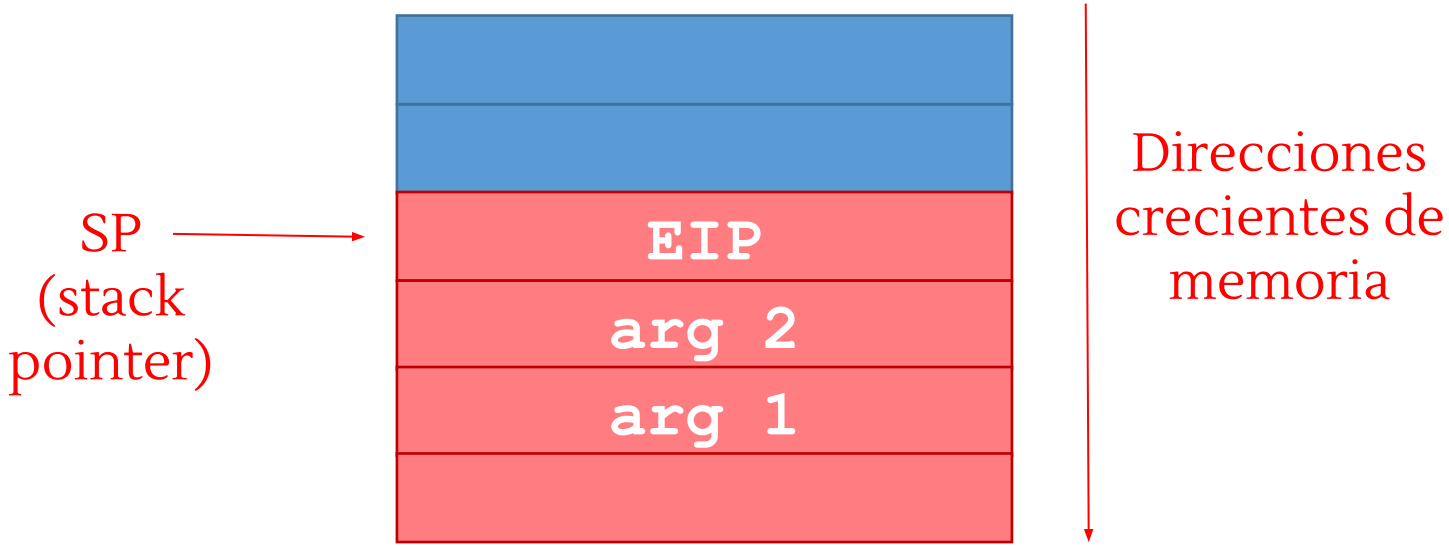
1) Previo al llamado a la función, se cargan los argumentos en la pila



Internals de llamados a funciones

Llamado a subrutina

2) Se carga el valor del contador de programa en la pila y se carga en el contador de programa la dirección de salto a la subrutina



Internals de llamados a funciones

Llamado a subrutina

3) Se ejecuta el código de la función y si devuelve un valor, se guarda el valor devuelto en el registro EAX

Más ejercicios

1) Escribir una función que reciba dos caracteres e indique si son iguales (1) o no (0) independientemente si están en mayúsculas o minúsculas.

```
int charComp(char c1, char c2);
```

2) Escribir una función que reciba un número y devuelva su factorial.

```
long int factorial (int num);
```

En ambos casos comprobar el correcto funcionamiento de las funciones.