Universidad Europea

LAUREATE INTERNATIONAL UNIVERSITIES

Paso de parámetros

Presentación

- Pocas funciones no necesitan comunicarse con el programa principal, para recoger valores iniciales o para devolver el resultado de su ejecución, a esto se le llaman Rutinas
- Las funciones aumenta su potencial enormemente, cuando intercambian información con el programa principal.
- La forma de intercambiar información:

Hasta ahora hemos visto funciones, una forma de encapsular código por funcionalidad.

Pero son pocas las funciones que no necesitan comunicarse con el exterior, normalmente para tomar valores con los que operar. Por ejemplo, una función que calcula qué día de la semana a la que corresponde una fecha cualquiera, necesita saber el día, mes y año de la fecha para calcular. Esos son parámetros de entrada.

A veces, las funciones también tiene parámetros de salida, como `por ejemplo, para el caso en cuestión, podría ser el día de semana correspondiente o incluso un error si por ejemplo es una fecha incorrecta.

Parámetros

El paso de parámetros entre el programa principal y las funciones se puede hacer de dos maneras.

A través de registros	A través de la Pila

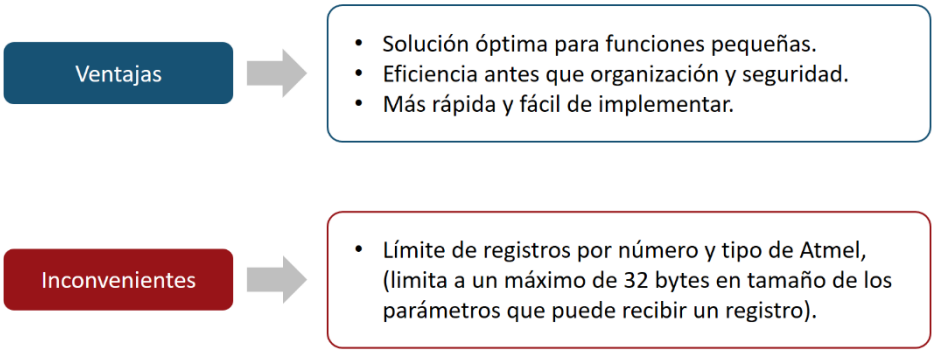
Es necesario establecer un orden y un protocolo, que permita saber:

- Cuántos parámetros.
- Tipo tamaño de parámetros se transmiten.
- Cuáles son de entrada y de salida.

Esta comunicación entre la función y el mundo exterior, puede ser a través de registros de microprocesador y a través de la pila.

Ambos métodos son buenos, pero necesitan establecer un protocolo.

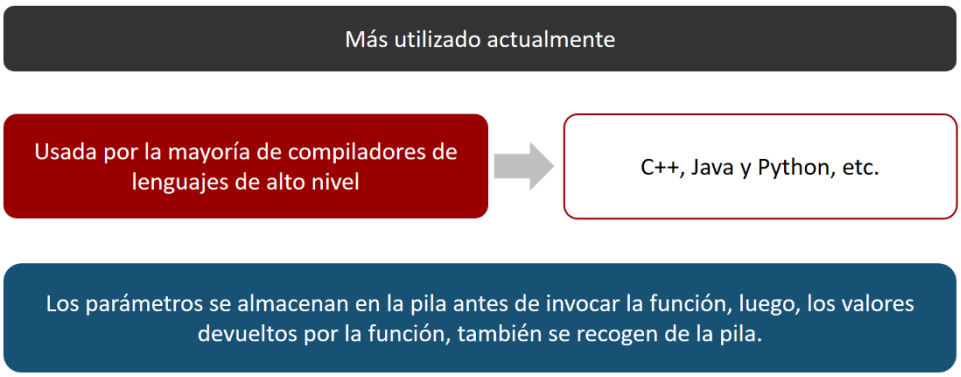
Paso de parámetros por registro



El paso de parámetros por registro, es muy rápida y efectiva, pero tiene limitaciones, principalmente que el número de registros es muy limitado y además, los registros, son un recurso hardware muy demandado y utilizado, como para bloquear su uso con los parámetros.

Pero su principal inconveniente es que el uso de registros hace un código poco portable y reutilizable, así como muy propenso a fallos colaterales, sobre todo por la modificación de registros que provocan fallos en tiempo de ejecución, muy difíciles de detectar y corregir.

Paso de parámetros por Pila (STACK)



El otro método es la utilización de la memoria PILA para paso de parámetros, no es tan eficiente, pero tiene las ventajas de que el tamaño se indefinido (lo limita la propia memoria del sistema).

Este es el método que se usa en la mayor parte de los lenguajes de programación (C, C++, java, etc.)

Los parámetros que se le pasan a una función, primero se almacenan en la pila y luego la función lee de la pila estos parámetros. Y el método es válido, tanto para parámetros de entrada como de salida.

Paso de parámetros por Pila (STACK)

Función que devuelve el positivo de una variable cualquiera.

Declaración (en C):	Invocación (en C):
• byte valor Absoluto (byte var);	• positivo = valor Absoluto (var);

La equivalencia en ensamblador de la invocación de esta función debería ser:

```
    ;Instrucciones anteriores a la llamada
    LDS R0, var          ;Cargar una de los parametros en un registro
    PUSH R0              ;Meter el contenido de la "var" en la pila.
    RCALL valorAbsoluto ;Llamar a la function valorAbsoluto
    POP R0               ;Sacar de la pila el resultado devuelto por la función
    STS positivo, R0     ;Almacenar el resultado en la variable correspondiente
    ;Instrucciones posteriores a la llamada
```

Pensemos en un ejemplo de paso de parámetros, como puede ser la llamada a la función de "ValorAbsoluto". Que en lenguaje C, es muy fácil, pero ¿cómo sería en ensamblador?.

El código correspondiente a la llamada sería este. Ojo, esta es la invocación a la función, no la función en sí misma.

Que analizado por fases, tenemos:

- Primero, pasar el parámetro a la función, es decir, guardamos el valor de la variable a calcular su "ValorAbsoluto", en la pila.

Recordemos que en Atmel, este proceso no puede ser directo, antes hay que pasar por un registro.

- Segundo, invocar la función "valorAbsoluto", instrucción CALL en Atmel.
- Tercero, sacar de la pila el parámetro de salida, que corresponde con el valor absoluto del parámetro de entrada.

Paso de más de un parámetro a una función

Ejemplo:

Supongamos una función que devuelve el mayor valor de dos variables, su declaración en C podría ser :

```
byte máximo (byte var1, byte var2)
```

Para invocar esta función en ensamblador de Atmel:

```
    ;Instrucciones anteriores a la llamada
    LDS R0, var1    ;Cargar uno de los parametros en un registro
    PUSH R0         ;Meter el contenido de la "var1" en la pila.
    LDS R0, var2    ;Cargar el segundo parametro en un registro
    PUSH R0         ;Meter el contenido de la "var2" en la pila.
    RCALL maximo    ;Llamar a la function maximo
    POP R0          ;Sacar de la pila el resultado devuelto por la función
    STS mayor, R0   ;Almacenar el resultado en la variable correspondiente
    POP R0          ;Sacar de la pila el parametro sobrante no utilizado
    ;Instrucciones posteriores a la llamada
```

¿Pero, y si tenemos que pasar más de un parámetro a la función?

La secuencia de pasos es muy parecida:

- Primero, insertar en pila todos los parámetros necesarios
- Segundo, llamada a la función
- Tercero, extraer el valor devuelto
- Cuarto paso, eliminar el exceso de parámetros y eliminarlos. Recordemos que esta función tiene dos parámetros de entrada y solo un de salida.

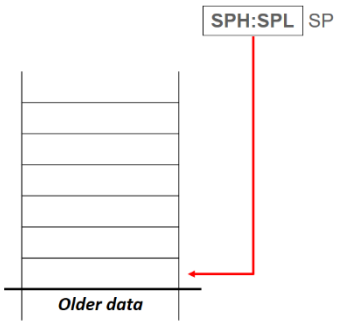
Algunos compiladores eliminan de pila los parámetros sobrantes de manera automática (como los compiladores para ordenadores Intel), pero el compilador de ATMEL no lo hace. De ahí que haya que eliminarlo manualmente.

Recepción de parámetros en una función

La pila evoluciona de la siguiente manera en una llamada a una función:

Paso de parámetros

```
LDS R0, var1
   PUSH R0    ;puts 1st parameter onto the stack
   LDS R0, var2
   PUSH R0    ;puts 2nd parameter onto the stack
```

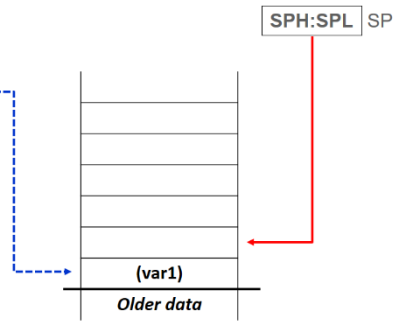


Recepción de parámetros en una función

La pila evoluciona de la siguiente manera en una llamada a una función:

Parámetro: var1

```
LDS R0, var1
   PUSH R0    ;puts 1st parameter onto the stack
   LDS R0, var2
   PUSH R0    ;puts 2nd parameter onto the stack
```

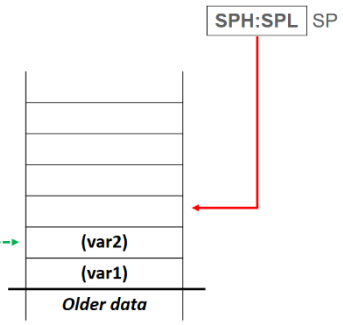


Recepción de parámetros en una función

La pila evoluciona de la siguiente manera en una llamada a una función:

Parámetro: var2

```
LDS R0, var1
   PUSH R0    ;puts 1st parameter onto the stack
   LDS R0, var2
   PUSH R0    ;puts 2nd parameter onto the stack
```

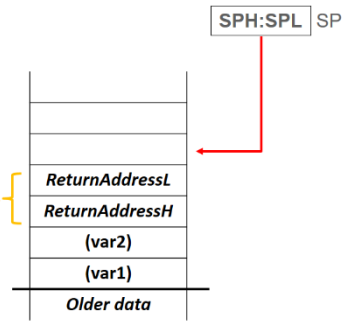


Recepción de parámetros en una función

La pila evoluciona de la siguiente manera en una llamada a una función:

Llamada a la función

```
LDS R0, var1
   PUSH R0    ;puts 1st parameter onto the stack
   LDS R0, var2
   PUSH R0    ;puts 2nd parameter onto the stack
   RCALL maximum ;calls the maximum function
```



Para visualizar esto, vamos a ver el ejemplo del valor absoluto, paso a paso, y de la evolución de la pila.

Para ello, partimos de un estado de pila inicial, que puede no estar vacía, como es este caso.

El registro SP apunta a la cima de la pila.

- Primero,

Se guarda en la pila, el primer parámetro de la función.

El registro SP se incrementa para seguir apuntando a la cima de la pila.

Es importante resaltar que realmente el registro SP se decrementa, ya que la pila empieza en valores de memoria altos y crece hacia los bajos.

- Segundo,

Se Inserta el segundo parámetro de la función en la pila

Nuevamente, el registro SP se vuelve a incrementar.

- Y para terminar,

Se llamada a la función.

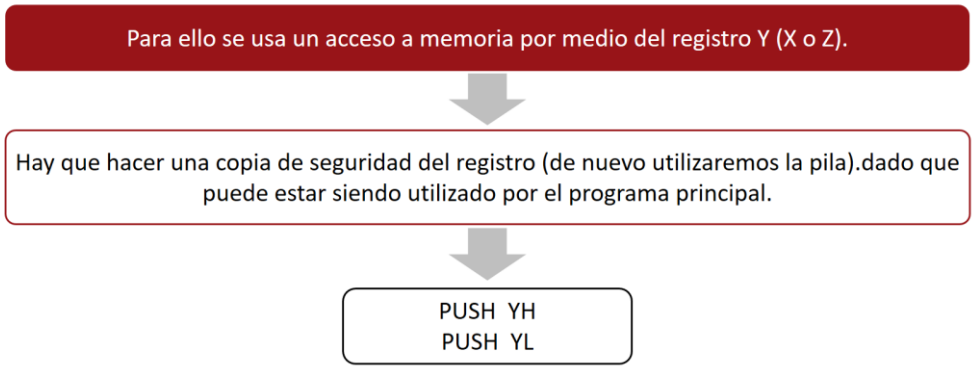
En la pila se guarda la dirección de retorno. Es decir, el valor almacenado en el registro PC, que indicaba cual sería la siguiente instrucción a ejecutar si no se hubiese llamado a la función, esta dirección, corresponde con la dirección de retorno.

Recordemos que las direcciones de memora del microprocesador Atmel son de 16bits, de ahí que ocupen dos espacios en pila.

Esto provoca que el registro SP se incremente en dos unidades.

Recepción de parámetros en una función

Los parámetros de la pila no se pueden extraer con un POP, porque la cima está ocupada por la dirección de retorno.



Hasta aquí, cómo pasar parámetros a una función a través de la pila, ahora, ¿cómo recuperar esos parámetros en la función?

No podemos utilizar las funciones POP que extrae un byte de la pila, ya que esta instrucción elimina el dato de la cima, por lo que eliminaría la dirección de retorno de la función, ya que es ese el dato que está en la cima de la pila.

La solución pasa por acceder directamente a esos valores en la pila, por medio de direcciones de memoria. La ventaja que tenemos, es que sabemos exactamente dónde están esos parámetros, por lo que su acceso es fácil y directo.

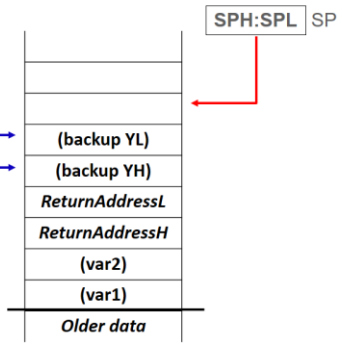
Además, recordemos que Atmel nos proporciona tres registros que están especialmente diseñados para acceder a memoria, los registros X, Y y Z.

Algo a tener cuidado. Si dentro de una función utilizamos un registro, debemos tener especial cuidado de salvaguardar su valor antes de utilizarlo, y restáuralo cuando terminemos de utilizarlo, ya que ese registro podría estar siendo utilizado en el programa principal. Y ¿dónde vamos a guardar el valor de esos registros que utilizaremos en la función? Pues en la propia pila.

Recepción de parámetros en una función

Instrucciones que tenemos que hacer dentro de nuestra función:

```
PUSH YH ;Guardamos la parte alta de Y en la pila
PUSH YL ;Guardamos la parte baja de Y en la pila
IN YL, SPL ;Inicializamos Y a SP: Parte alta
IN YH, SPH ;Inicializamos Y a SP: Parte baja
   ;Ahora Y es usado como base de puntero a datos.
```

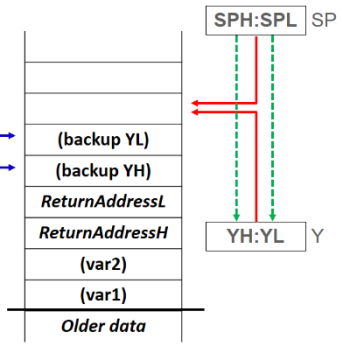


Ya dentro de la función, lo primero que tenemos que hacer, es guardar el actual valor del registro puntero que vamos utilizar para direccionar la memoria donde están los parámetros de la función. En este caso, vamos a utilizar el registro Y: Instrucciones PUSH

Recepción de parámetros en una función

Instrucciones que tenemos que hacer dentro de nuestra función:

```
PUSH YH ;Guardamos la parte alta de Y en la pila
PUSH YL ;Guardamos la parte baja de Y en la pila
IN YL, SPL ;Inicializamos Y a SP: Parte alta
IN YH, SPH ;Inicializamos Y a SP: Parte baja
   ;Ahora Y es usado como base de puntero a datos.
```



Y luego inicializar el registro Y al valor actual del registro SP. Esto permite hacer una foto del estado actual de la pila. A partir de este momento, aunque se inserten mas valores en la pila lo que provocaría la modificación del registro SP, nosotros tenemos los parámetros localizados de manera exacta desde el registro Y, ya que este permanecerá inalterado, manteniéndonos una instantánea estable de la pila del momento de la invocación a la función.

Es decir, tanto SP como Y apuntaran a la actual cima de la pila.

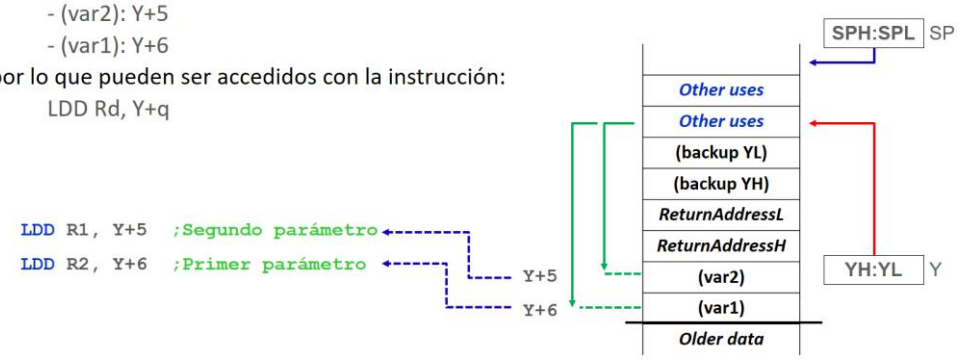
Recepción de parámetros en una función

Una vez inicializado, tenemos un punto estable para acceder a los datos, aunque la pila se modifique.
Los parámetros, están accesibles en la direcciones:

- (var2): Y+5
- (var1): Y+6

por lo que pueden ser accedidos con la instrucción:

LDD Rd, Y+q



Después de hacer la instantánea de la pila, ya podemos acceder a los parámetros. Que están localizables a través del registro Y teniendo en cuenta un desplazamiento.

Es decir, el primer parámetro esta 5 bytes mas abajo de la dirección apuntado por Y (sumar 5), el segundo parámetro 6 posiciones mas abajo, y así sucesivamente para todos los parámetros.

Y esto se cumple, aunque hayamos movido el registro SP por haber insertado mas valores en la pila.

Devolución de parámetros en una función

Para devolver valores, se utiliza uno de los parámetros de entrada como salida.

- Sobrescribiendo el parámetro.
- Empezando siempre con el primer parámetro.
- Normalmente una función solo devuelve un parámetro.
  pero si se necesitasen devolver mas de un parámetro, hay que forzar a que la función tenga al menos tantos parámetros de entrada como salida. Creando parámetros de entrada vacíos.

El acceso a los parámetros de salida es muy parecido a los de entrada. La instrucción utilizada es:

```
STD Y+q, Rd
```

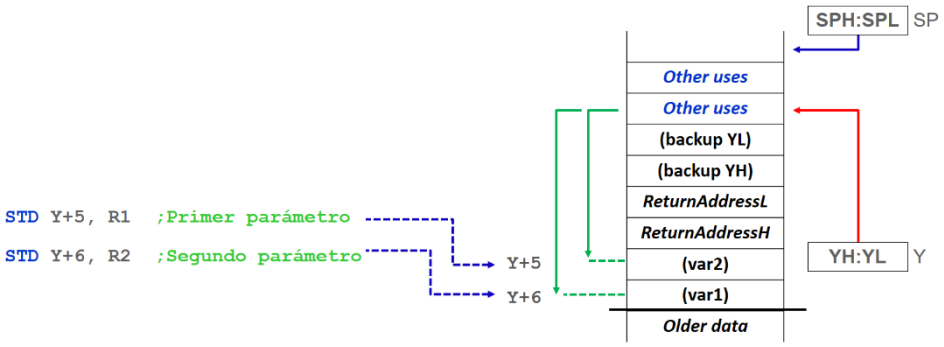
Una vez hemos recogido los parámetros de la pila en la función, ya podemos operar con ellos.

Después, debemos devolver los valores calculados en la función, es decir, devolver los parámetros de salida al Programa Principal

Para ello, una vez más, utilizaremos la pila, reutilizando los propios parámetros de entrada, para almacenar los parámetros de salida.

Devolución de parámetros en una función

Ejemplo:

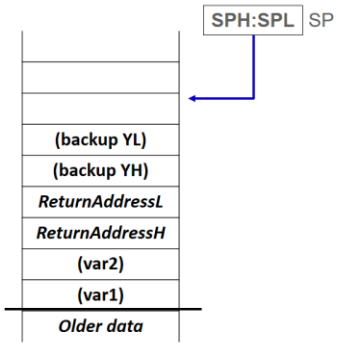


Por Ejemplo, esta sería el método de devolución de parámetros desde la función.

Como aún tenemos el registro Y estable, al igual que hemos recogido los parámetros de entrada, ahora podemos utilizar la función STD para guardar las parámetros de salida, ya que esta instrucción permite mover datos de registro a memoria

Ejemplo de parámetros en una función

```
   ; Cabecera: byte máximo (byte var1, byte var2)
   ; Devuelve el mayor de dos parámetros pasados por pila
maximo: ;Inicio de la función
   PUSH YH       ;backup del reg. Y,
   PUSH YL       ;  parte alta y baja.
   IN YL, SPL    ;Iniciamos el reg.
   IN YH, SPH    ;  a la cima de la pila
   PUSH R1       ;backup R1
   PUSH R0       ;backup R0
   LDD R1, Y+5   ;sacamos var2 de la pila
   LDD R0, Y+6   ;Sacamos var1 de la pila
   CP R0, R1     ;comparamos ambos valores
   BRLT terminar; Si var2 es mayor Terminaos. Ya que el
                 ;   valor ya esta bien guardado en la 1° posición.
   STD Y+5, R0   ;Si el mayor es var1, lo guardamos en el 1°
                 ;   parámetros de pila
terminar:
   POP R0        ;Restauramos R0 antes de terminar.
   POP R1        ;Restauramos R1 antes de terminar.
   POP YL        ;Restauramos YL antes de terminar.
   POP YH        ;Restauramos YH antes de terminar.
   RET
```

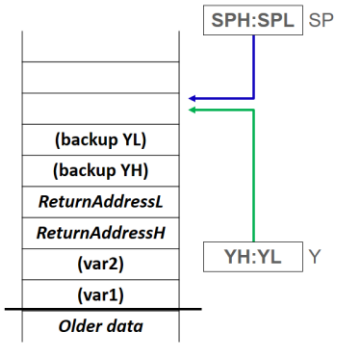


Recopilando:

El ejemplo completo de la estructura de una función incluye:

Ejemplo de parámetros en una función

```
    ; Cabecera: byte máximo (byte var1, byte var2)
    ; Devuelve el mayor de dos parámetros pasados por pila
maximo: ;Inicio de la función
start PUSH YH       ;backup del reg. Y,
      PUSH YL       ;  parte alta y baja.
      IN YL, SPL    ;Iniciamos el reg.
      IN YH, SPH    ;  a la cima de la pila
      PUSH R1       ;backup R1
      PUSH R0       ;backup R0
      LDD R1, Y+5   ;sacamos var2 de la pila
      LDD R0, Y+6   ;Sacamos var1 de la pila
      CP R0, R1     ;comparamos ambos valores
      BRLT terminar; Si var2 es mayor Terminaos. Ya que el
                    ;   valor ya esta bien guardado en la 1° posición.
      STD Y+5, R0   ;Si el mayor es var1, lo guardamos en el 1°
                    ;   parámetros de pila
terminar:
      POP R0        ;Restauramos R0 antes de terminar.
      POP R1        ;Restauramos R1 antes de terminar.
      POP YL        ;Restauramos YL antes de terminar.
      POP YH        ;Restauramos YH antes de terminar.
      RET
```



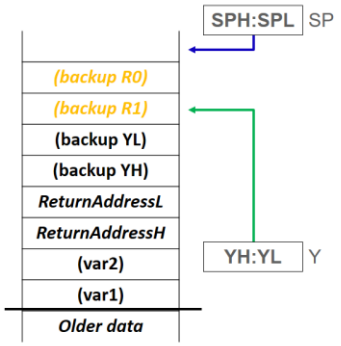
- Tomar la foto del estado actual de la pila, para tener la referencia exacta a los parámetros de entrada.
- Que a su vez, consta de las opciones del backup del registro Y, y de la copia del registro SP en Y

Ejemplo de parámetros en una función

```
    ; Cabecera: byte máximo (byte var1, byte var2)
    ; Devuelve el mayor de dos parámetros pasados por pila
maximo: ;Inicio de la función
  PUSH YH       ;backup del reg. Y,
  PUSH YL       ;  parte alta y baja.
  IN YL, SPL    ;Iniciamos el reg.
  IN YH, SPH    ;  a la cima de la pila
  PUSH R1       ;backup R1
  PUSH R0       ;backup R0
  LDD R1, Y+5   ;sacamos var2 de la pila
  LDD R0, Y+6   ;Sacamos var1 de la pila
  CP R0, R1     ;comparamos ambos valores
  BRLT terminar; Si var2 es mayor Terminaos. Ya que el
                ;   valor ya esta bien guardado en la 1° posición.
  STD Y+5, R0   ;Si el mayor es var1, lo guardamos en el 1°
                ;   parámetros de pila
terminar:
  POP R0        ;Restauramos R0 antes de terminar.
  POP R1        ;Restauramos R1 antes de terminar.
  POP YL        ;Restauramos YL antes de terminar.
  POP YH        ;Restauramos YH antes de terminar.
  RET
```

start

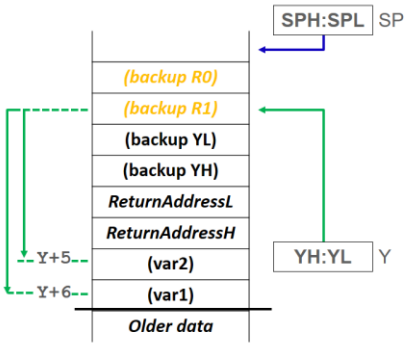
Regs saving



- Backup de los registros utilizados en la función. Al igual que se hace con el registro Y, hay que hacerlo con todos los registros utilizados en la función. En este caso, los registros r0 y r1, ya que son los únicos registro utilizados dentro de la función.

Ejemplo de parámetros en una función

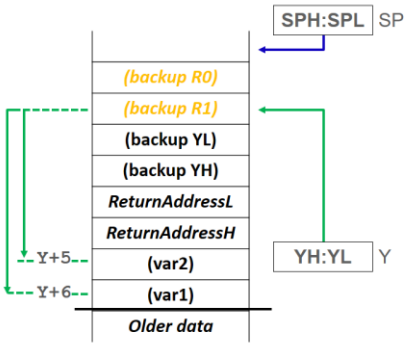
```
        ; Cabecera: byte máximo (byte var1, byte var2)
        ; Devuelve el mayor de dos parámetros pasados por pila
maximo: ;Inicio de la función
start   PUSH YH       ;backup del reg. Y,
        PUSH YL       ;  parte alta y baja.
        IN YL, SPL    ;Iniciamos el reg.
        IN YH, SPH    ;  a la cima de la pila
Regs    PUSH R1       ;backup R1
saving  PUSH R0       ;backup R0
Parameter LDD R1, Y+5   ;sacamos var2 de la pila
        LDD R0, Y+6   ;Sacamos var1 de la pila
        CP R0, R1     ;comparamos ambos valores
        BRLT terminar; Si var2 es mayor Terminaos. Ya que el
                      ;   valor ya esta bien guardado en la 1° posición.
        STD Y+5, R0   ;Si el mayor es var1, lo guardamos en el 1°
                      ;   parámetros de pila
terminar:
        POP R0        ;Restauramos R0 antes de terminar.
        POP R1        ;Restauramos R1 antes de terminar.
        POP YL        ;Restauramos YL antes de terminar.
        POP YH        ;Restauramos YH antes de terminar.
        RET
```



- Acceso a los parámetros de entrada a la función, utilizando en registro Y, y un desplazamiento.

Ejemplo de parámetros en una función

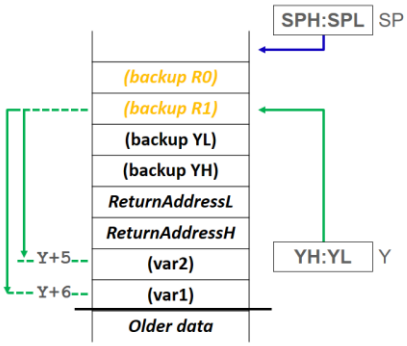
```
        ; Cabecera: byte máximo (byte var1, byte var2)
        ; Devuelve el mayor de dos parámetros pasados por pila
maximo: ;Inicio de la función
start   PUSH YH       ;backup del reg. Y,
        PUSH YL       ;  parte alta y baja.
        IN YL, SPL    ;Iniciamos el reg.
        IN YH, SPH    ;  a la cima de la pila
Regs    PUSH R1       ;backup R1
saving  PUSH R0       ;backup R0
Parameter LDD R1, Y+5 ;sacamos var2 de la pila
        LDD R0, Y+6   ;Sacamos var1 de la pila
body    CP R0, R1     ;comparamos ambos valores
        BRLT terminar; Si var2 es mayor Terminaos. Ya que el
                      ;   valor ya esta bien guardado en la 1° posición.
        STD Y+5, R0   ;Si el mayor es var1, lo guardamos en el 1°
                      ;   parámetros de pila
terminar:
        POP R0        ;Restauramos R0 antes de terminar.
        POP R1        ;Restauramos R1 antes de terminar.
        POP YL        ;Restauramos YL antes de terminar.
        POP YH        ;Restauramos YH antes de terminar.
        RET
```



- Cuerpo de la función, es el trozo de código que básicamente tendremos que implementar en nuestra función.

Ejemplo de parámetros en una función

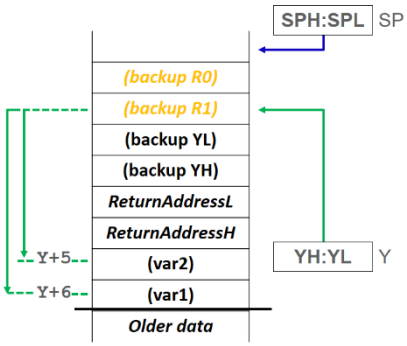
```
; Cabecera: byte máximo (byte var1, byte var2)
; Devuelve el mayor de dos parámetros pasados por pila
maximo: ;Inicio de la función
  PUSH YH       ;backup del reg. Y,          start
  PUSH YL       ;  parte alta y baja.
  IN YL, SPL    ;Iniciamos el reg.
  IN YH, SPH    ;  a la cima de la pila
  PUSH R1       ;backup R1                   Regs saving
  PUSH R0       ;backup R0
  LDD R1, Y+5   ;sacamos var2 de la pila     Parameter
  LDD R0, Y+6   ;Sacamos var1 de la pila
  CP R0, R1     ;comparamos ambos valores    body
  BRLT terminar; Si var2 es mayor Terminaos. Ya que el
                ;   valor ya esta bien guardado en la 1° posición.
  STD Y+5, R0   ;Si el mayor es var1, lo guardamos en el 1°  Return Parameter
                ;   parámetros de pila
terminar:
  POP R0        ;Restauramos R0 antes de terminar.
  POP R1        ;Restauramos R1 antes de terminar.
  POP YL        ;Restauramos YL antes de terminar.
  POP YH        ;Restauramos YH antes de terminar.
  RET
```



- Modificación de los parámetros de entrada, con los valores de salida.

Ejemplo de parámetros en una función

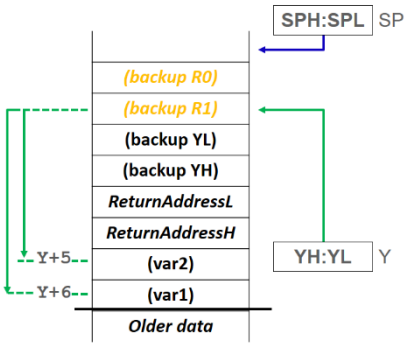
```
        ; Cabecera: byte máximo (byte var1, byte var2)
        ; Devuelve el mayor de dos parámetros pasados por pila
maximo: ;Inicio de la función
start   PUSH YH       ;backup del reg. Y,
        PUSH YL       ;  parte alta y baja.
        IN YL, SPL    ;Iniciamos el reg.
        IN YH, SPH    ;  a la cima de la pila
Regs    PUSH R1       ;backup R1
saving  PUSH R0       ;backup R0
Parameter LDD R1, Y+5 ;sacamos var2 de la pila
        LDD R0, Y+6   ;Sacamos var1 de la pila
body    CP R0, R1     ;comparamos ambos valores
        BRLT terminar ; Si var2 es mayor Terminaos. Ya que el
                      ;   valor ya esta bien guardado en la 1° posición.
Return  STD Y+5, R0   ;Si el mayor es var1, lo guardamos en el 1°
Parameter             ;   parámetros de pila
terminar:
Regs    POP R0        ;Restauramos R0 antes de terminar.
recovering POP R1     ;Restauramos R1 antes de terminar.
        POP YL        ;Restauramos YL antes de terminar.
        POP YH        ;Restauramos YH antes de terminar.
        RET
```



- Restaurar los valores de r0 y r1 que previamente habíamos almacenado en la pila.

Ejemplo de parámetros en una función

```
        ; Cabecera: byte máximo (byte var1, byte var2)
        ; Devuelve el mayor de dos parámetros pasados por pila
maximo: ;Inicio de la función
start   PUSH YH       ;backup del reg. Y,
        PUSH YL       ;  parte alta y baja.
        IN YL, SPL    ;Iniciamos el reg.
        IN YH, SPH    ;  a la cima de la pila
Regs    PUSH R1       ;backup R1
saving  PUSH R0       ;backup R0
Parameter LDD R1, Y+5 ;sacamos var2 de la pila
        LDD R0, Y+6   ;Sacamos var1 de la pila
body    CP R0, R1     ;comparamos ambos valores
        BRLT terminar ; Si var2 es mayor Terminaos. Ya que el
                      ;   valor ya esta bien guardado en la 1° posición.
Return  STD Y+5, R0   ;Si el mayor es var1, lo guardamos en el 1°
Parameter             ;   parámetros de pila
terminar:
Regs    POP R0        ;Restauramos R0 antes de terminar.
recovering POP R1     ;Restauramos R1 antes de terminar.
        POP YL        ;Restauramos YL antes de terminar.
end     POP YH        ;Restauramos YH antes de terminar.
        RET
```



- Restaurar el registro Y, y fin de la función, retornando al punto de llamada

Variables locales

Los compiladores modernos, utilizan la pila como lugar para crear las variables locales de una función.

Procedimiento muy eficiente y libera de errores al programador. Incrementa manualmente el registro SP, desplazando la cima de la pila los bytes necesarios para las variables locales.

Importante:
Restaurar la cima de la pila, al terminar de usar las variables locales.

En los compoiladores modernos, este uso de la pila, se lleva al extremo, por ejemplo, creando en pila las variables locales a una función.

Variables locales

- Ejemplo para crear un espacio para cuatro variables byte en la pila.

– Reserve de espacio

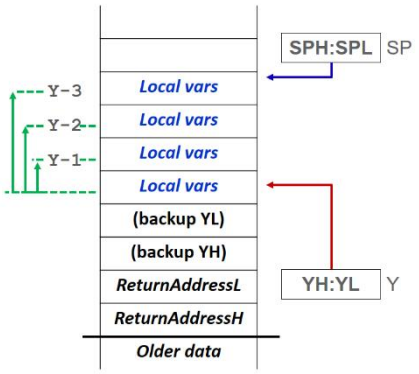
```
MOV R16, YL  ;Copiar Y el los registros [r16:r17]
MOV R17, YH
SUBI R16,4   ;Restar 4 al registro [r16:r17]
SBCI R17,0
OUT SPL, R16 ;Asignar a SP el valor desplazado
OUT SPH, R17
```

– Ya tenemos el espacio creado. Accesible con:

```
Y, Y-1, Y-2, Y-3
```

– Antes de retornar, hay que devolver el espacio:

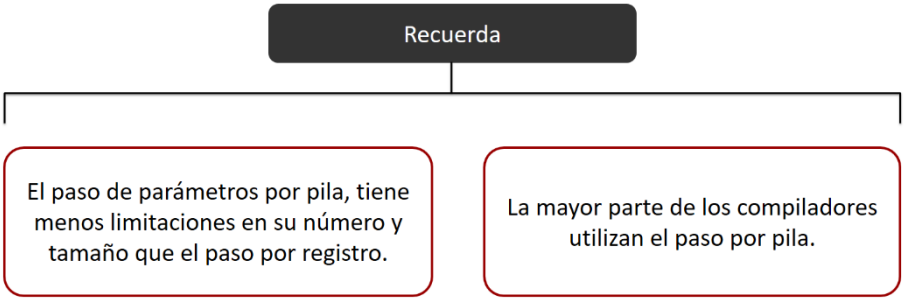
```
OUT SPL, YL
OUT SPH, YH  ;Ponemos SP en l posición original
POP YL
POP YH       ;Restauramos los valores de Y
RET
```



El principio de funcionamiento es el mismo, al crear una variable, lo que realmente se hace es reservar espacio en la pila. Como esta reserva se realiza dentro de la función, el registro Y ya ha tomado una instantánea de la pila, por lo que estas variables, están accesibles del mismo modo que los parámetros de la función, pero esta vez, teniendo en cuenta que estas variables están mas arriba de Y, por lo que sus direcciones relativas, se consiguen decrementeando el valor de Y.

Pero esto, es solo información interesante de como funcionan los compiladores y lenguajes de programación de alto nivel. En Atmel, no tendremos que utilizar estas técnicas. Pero nos da una idea del gran uso que se le da a la pila en los lenguajes de alto nivel.

Variables locales



Para terminar, solo recordaros, de que aunque el paso de parámetros a una función se pueden hacer de varias maneras, y a pesar de que el paso por registro es muy rápido y eficiente, es poco independiente de contexto y muy propenso a errores colaterales.

En la actualidad, la pila es el método más utilizado, y el que nosotros vamos a utilizar en esta asignatura.