Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2343 – Arquitectura de Computadores

Arquitectura x86 – Convenciones de llamada

Profesor: Hans Löbel

Uso de subrutinas en x86 presenta mayor complejidad que en el computador básico

- En nuestro computador, el stack almacenaba la dirección de retorno.
- No había una convención sobre dónde almacenar los parámetros y valores de retorno.
- En x86, utilizaremos el stack de manera más explícita: parámetros, retorno, variables locales.
- Uso del registro BP (base pointer) es fundamental para facilitar el manejo de todos estos datos.

Uso de subrutinas requiere la definición de convenciones de llamada (*calling conventions*)

- Las convenciones definen la interfaz sobre la cual trabajará el código de la subrutina.
- Una convención de llamada debe especificar lo siguiente:
  - Donde se encuentran los parámetros (stack, registros o una mezcla de ambos).
  - Si se usa el stack, el orden en que los parámetros son entregados.
  - Definición de responsabilidades de restauración del stack, entre la subrutina y el código que la llama.

Uso de subrutinas requiere la definición de convenciones de llamada (*calling conventions*)

- Existen múltiples convenciones: *stdcall*, *cdecl*, *fastcall, safecall, syscall, thiscall,…*
- Se dividen entre las que asignan la responsabilidad de limpieza del stack a la subrutina, y las que se la asignan al código que llama a la subrutina.
- Ocuparemos la convención *stdcall*, que es la usada por la API Win32 de Microsoft.

stdcall deposita en la subrutina la responsabilidad de limpiar el stack

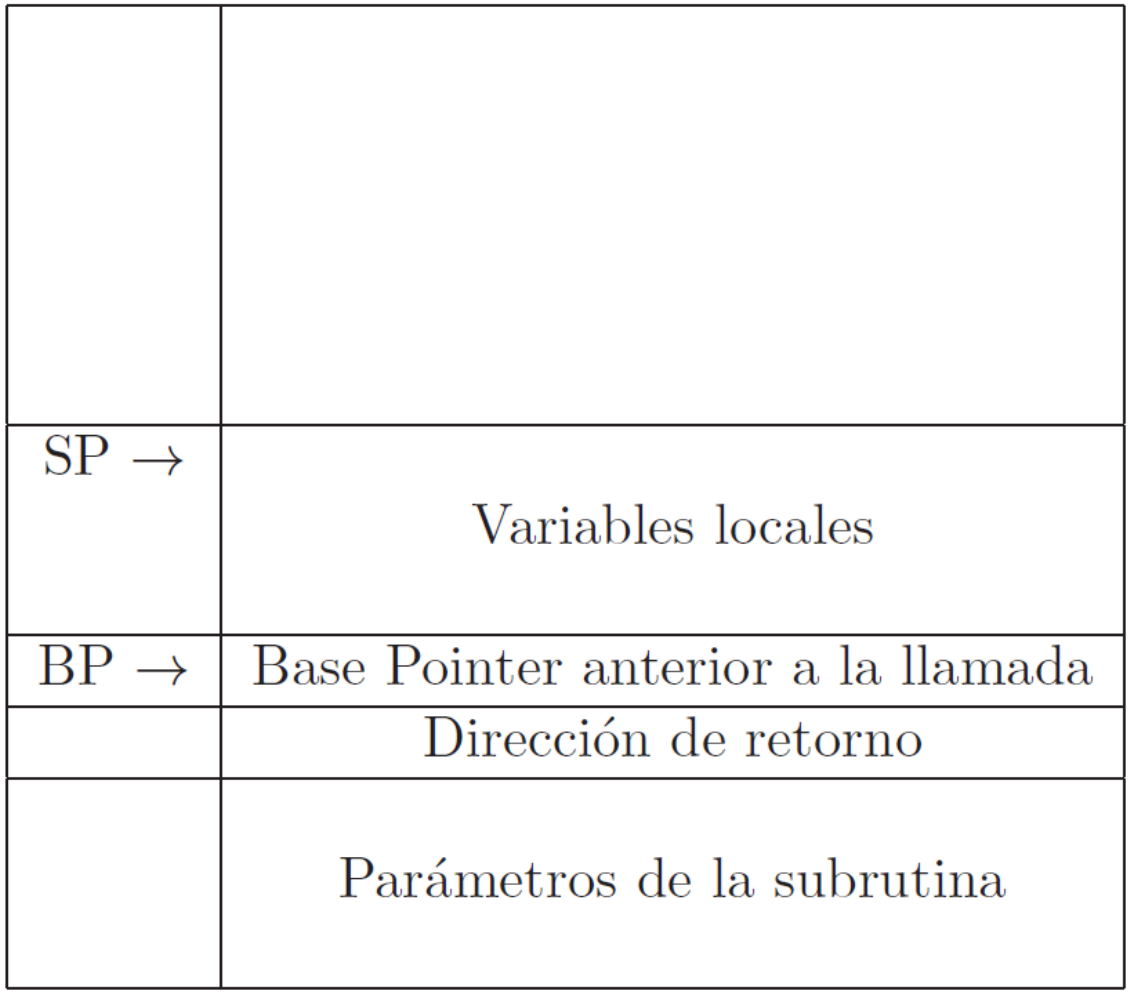
La convención *stdcall* especifica los siguientes 3 puntos:

1. Los parámetros son pasados de derecha a izquierda, usando el stack.

2. El retorno se almacenará en el registro AX

3. La subrutina se debe encargar de dejar SP apuntando en la misma posición que estaba antes de pasar los parámetros.

En *stdcall*, SP y BP permiten tener llamadas anidadas de subrutinas (recursión) y variables locales



Se deben ejecutar 2 pasos al momento de llamar a una subrutina

1. Agregar los parámetros al stack usando la instrucción PUSH. Los valores agregados sólo pueden ser de 16 bits.

2. Llamar a la subrutina con la instrucción CALL, lo que almacena en el stack la dirección de retorno y ejecuta el salto a la dirección de la subrutina.

Dentro de la subrutina, son 5 los pasos a ejecutar

1. Guardar el valor actual de BP en el stack y cargar el valor de SP en BP:

   PUSH BP

   MOV BP, SP

   En caso de usar variables locales, se debe reservar espacio para estas, moviendo SP *n* posiciones hacia arriba, donde *n* es el número de palabras de memoria que usan las variables:

   SUB SP, n

Dentro de la subrutina, son 5 los pasos a ejecutar

2. Ejecutar la subrutina. Para acceder a los parámetros se usa direccionamiento mediante BP. Así, el primer parámetro estará en BP+4, el segundo en BP+6, etc.

   De la misma manera, las variables locales se acceden usando BP, pero con offset negativo, BP-2, etc.

3. Al finalizar la subrutina se debe recuperar el espacio de las variables locales:

   ADD SP, n

Dentro de la subrutina, son 5 los pasos a ejecutar

4. Rescatar el valor previo de BP:

   POP BP

5. Mover SP al valor previo al paso de los parámetros:

   RET *n*

   donde *n* indica la cantidad de palabras de memoria usadas por los parámetros.

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2343 – Arquitectura de Computadores

Arquitectura x86 – Convenciones de llamada

Profesor: Hans Löbel