Interacción C-Assembler

Organización del Computador II

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

2do cuatrimestre de 2006

1. Cómo empezar y terminar una función en assembler

Es costumbre (y es recomendable) que cada función defina su propio espacio en la pila. De esta manera los parámetros con los que se llamó a la función quedan claramente separados respecto de la pila local, en la que además se puede reservar espacio para variables locales.

Generalmente esto se hace de la siguiente manera:

```
funcion:
  push ebp     ; salva la base de la pila anterior
  mov ebp, esp ; reposiciona la nueva base en el tope de la pila
  sub esp, 8   ; reserva 2 dword de espacio para variables locales
  ...
  pop ebp      ; recupera la base anterior
  ret
```

2. Cómo se pasan parámetros y se devuelven resultados en convención C

En el lenguaje C, los parámetros de una función se pasan por pila, pushandolos desde el último hasta el primero, en ese orden. Es decir, para una función de prototipo

`int func(int a, int b, int c);`

se pushea primero el valor correspondiente a `c`, luego el de `b` y por último el de `a`.

Luego de reposicionar la base de la pila moviendo el tope de la pila (`esp`) a la base de la pila (`ebp`), el primer parámetro queda en `[ebp+8]`, el segundo en `[ebp+12]`, y así sucesivamente. Notar que en `[ebp+4]` está la dir de retorno que fue pushada por el call.

La función que llama debe ocuparse de sacar los parámetros de la pila después de la llamada. La función llamada pudo haber modificado los parámetros pasados, por lo tanto no se puede suponer que mantienen los datos originales.

Por otro lado, las funciones que acuerdan con la convención C devuelven por medio del registro `eax`.

Además, por convención C, se deben salvar los siguientes registros: `esi`, `edi`, `ebx`. En realidad, estos registros deben salvarse explícitamente si son usados, pero también deben salvarse

los registros asociados al manejo de la pila (`esp` y `ebp`) pues al finalizar la función la pila debe quedar en el mismo estado.

3. Cómo hacer para usar desde C una función escrita en assembler

Desde C:

Sintaxis: `extern <prototipo de la funci'on>;`

El keyword "`extern`" indica que la función está definida en algún otro lugar fuera del módulo C.

Ejemplo: `extern void funcionAsm(int n);`

Desde Assembler:

Sintaxis: `global <definicion del simbolo>`

"`global`" le indica al ensamblador que este símbolo se incluya en la tabla de símbolos globales para que cualquier otro módulo lo pueda ver.

Ejemplo:

```
global funcionAsm
...
FuncionAsm:
  ; aqui va el codigo para FuncionAsm
  ret
```

4. Cómo hacer para usar una función C desde assembler

Desde Assembler:

Sintaxis: `extern <funcionC>`

La directiva "`extern`" indica que la función está definida en algún otro lugar fuera del módulo Assembler.

Ejemplo:

```
extern funcionC
...
call funcionC
```

5. Ejemplo

Archivo `.c`:

```
extern int funcionAsm(char* param1, int param2);

int main(void){
  return funcionAsm("Orga",2);
}
```

Archivo `.asm`**:**

```
global funcionAsm

extern printf

section .data
formato db '%s %d',10,0

section .text
funcionAsm:
  push ebp
  mov ebp,esp
  push esi
  push edi
  push ebx

  push dword [ebp+12] ;param2
  push dword [ebp+8] ;param1
  push dword formato
  call printf
  add esp,12
  xor eax,eax ;devuelve 0

  pop ebx
  pop edi
  pop esi
  pop ebp
  ret
```