MIPS ABI: Function Calling Convention
Organización de computadoras - 66.20

1. Alcance de este documento

Este documento no pretende ser una descripción completa de la ABI presentada en [1], sino un detalle de los puntos que se consideran confusos, más importantes o que varían respecto al documento mencionado.

Este documento se basa en [1] y en las observaciones realizadas sobre código assembly generado con GCC, con nivel de optimización 0 ($-O0$).

Por lo tanto, se recomienda *primero* leer la sección *"Function Calling Sequence"* de [1], para luego aclarar y/o modificar con el contenido de este documento.

2. Caller y Callee Saved Registers

Los siguientes registros deben ser salvados por la función llamada si ésta los debe modificar:

- ra, sp, fp (o s8), gp, s0 ... s7
- f20 a f30 (floating point)

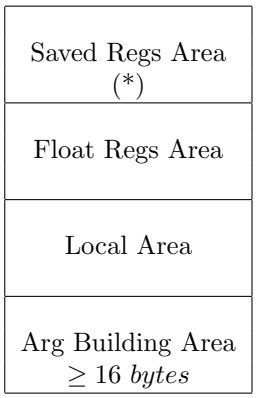
El resto de los registros no se garantiza que su valor se preserve entre llamadas a funciones. Si la función caller necesita preservarlos, debe salvarlos ella en su stack frame.

3. Stack Frame

- Cada funcion crea su *Stack Frame* siempre [1];
- El *Stack Frame* se compone de *areas* de un tamaño múltiplo de 8 bytes, alineadas a 8 bytes [2];
- Los registros en un area se almacenan de abajo hacia arriba, en orden según el número de registro.
- Las areas son, en orden de aparición (de arriba hacia abajo en memoria):

[1] según lo observado con GCC, [1] indica que sólo cuando es necesario

[2] según lo observado con GCC, [1] lo aplica a algunas areas y agrega padding para la alineación a 8 bytes del stack frame completo



Cuadro 1: Layout genérico del stack frame. Las areas marcadas con (*) son obligatorias.

- General Register Save Area (obligatoria [3]). Registros salvados:
  - Siempre: fp, gp;
  - Cuando la función es non-leaf: ra;
  - Si es necesario: el resto de los *general purpose callee-saved regs.*
- Floating-Point Registers Save Area: f20 a f30 si es necesario (ver 3-15 de [1] para más detalle) [4];
- Local and Temporary Variables Area;
- Argument Building Area.
  - Se crea cuando la función es non-leaf
  - Al menos es de 16 bytes, aún cuando los argumentos del callee requieren menos.
  - Cuando los cuatro primeros argumentos son enteros o punteros, se pasan en $a0$ a $a3$, y el resto se guarda en esta area a partir de los primeros 16 bytes.
  - Los argumentos pasados en $a0$ a $a3$ son almacenados en esta area por el callee[5] siempre [6].

- Debugging: algunas reglas de creación de stack frame existen sólo para garantizar que las herramientas de debugging (ej: gdb) puedan puedan realizar un stack backtrace (es decir, algunas reglas pueden no seguirse y el programa funcionará correctamente, pero no se garantiza un correcto analisis post-mortem en caso de que el proceso termine de forma anormal).

[3] no es obligatoria en [1] por lo dicho antes, pero GCC siempre la crea
[4] no pudo ser verificado
[5] sí, por el callee, es el único caso en el cual el callee escribe en el stack frame del caller
[6] nuevamente, es lo observado con GCC, no se especifica esto en [1]

fp ($30)
gp ($28)

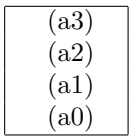
Cuadro 2: Ejemplo de Saved Regs Area mínima – función leaf.

alignment [4]
ra ($31)
fp ($30)
gp ($28)

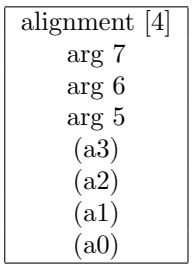
Cuadro 3: Ejemplo de Saved Regs Area mínima para una función non-leaf. También puede salvar el resto de los callee-saved registers.

Referencias

[1] "System V Application Binary Interface", MIPS RISC Processor, 3rd Edition, The Santa Cruz Operation, February 1996 (`http://www.sco.com/developers/devspecs/mipsabi.pdf`).



Cuadro 4: Ejemplo de Argument Building Area (a0 . . . a3 son salvados por el callee) cuando la función a llamar tiene hasta 4 argumentos enteros y/o punteros.



Cuadro 5: Ejemplo de Argument Building Area (a0 . . . a3 son salvados por el callee) cuando la función a llamar tiene 7 argumentos enteros y/o punteros.