Práctica 5: Lenguaje Ensamblador

Orea Romero Laura Victoria
Velasco Gutierrez Fernando

Pregunta 1

¿Qué hacen las instrucciones de tipo FR y de tipo FI? Da algunos ejemplos de instrucciones de este tipo y menciona por qué están separadas de las otras 3 principales.

En la arquitectura MIPS, las instrucciones se clasifican en varios formatos. Los tipos **FR** (Formato de Registro Flotante) y **FI** (Formato Inmediato Flotante) están relacionados con operaciones de punto flotante.

- **Tipo FR (Floating-point Register):** Son instrucciones que operan exclusivamente con registros de punto flotante. Algunos ejemplos son:
  - `add.s $f0, $f1, $f2` (Suma de precisión simple)
  - `sub.d $f4, $f6, $f8` (Resta de precisión doble)
  - `mul.s $f10, $f12, $f14` (Multiplicación de precisión simple)
- **Tipo FI (Floating-point Immediate):** Son instrucciones que incluyen un valor inmediato (constante) en la operación con registros de punto flotante.

Estas instrucciones están separadas de los otros tres formatos principales (Tipo R, I, J) porque:

- Usan registros y formatos específicos para punto flotante.
- Tienen su propio conjunto de registros ($f0 a $f31).
- Las operaciones flotantes requieren manejo de excepciones y formatos numéricos distintos.

Pregunta 2

¿Qué es la Portabilidad de Arquitecturas (Cross-Architecture Porting)?

La **Portabilidad de Arquitecturas** se refiere a la capacidad de adaptar software compilado o escrito para una arquitectura específica (como x86, ARM, MIPS) para que funcione en otra arquitectura diferente, sin reescribir completamente el código.

Por ejemplo:
Portar un sistema operativo como Linux de x86 a ARM, o una aplicación de MIPS a RISC-V.

Técnicas comunes:

- Recompilación del código fuente para la nueva arquitectura.
- Uso de emuladores o máquinas virtuales.
- Traducción binaria.

Pregunta 3

¿Qué son las arquitecturas x86 y x64? ¿Tienen un único lenguaje ensamblador?

- **x86:** Es una arquitectura de 32 bits desarrollada por Intel, usada en la mayoría de las PC desde los 80s, esta basada en el conjunto de instrucciones CISC.
- **x64 (x86-64):** Es una extensión de 64 bits de x86, mantiene compatibilidad hacia atrás pero amplía registros y capacidades.

No tienen un único lenguaje ensamblador. Aunque el conjunto de instrucciones es el mismo, existen varios **sintaxis** de lenguaje ensamblador para x86/x64, como:

- **Intel Syntax**: Usada en documentos de Intel y Microsoft.
- **AT&T Syntax**: Usada en GNU (GCC, GAS).

Pregunta 4

En el contexto de los lenguajes de ensamblador, ¿Qué es NASM? ¿Qué es MASM?

- **NASM (Netwide Assembler):** Ensamblador libre y gratuito para la arquitectura x86/x64. Soporta sintaxis Intel y es multiplataforma (Linux, Windows, macOS). Muy usado en desarrollo de sistemas y hacking low-level.
- **MASM (Microsoft Macro Assembler):** Ensamblador desarrollado por Microsoft para x86/x64. Usado principalmente en entornos Windows y con herramientas de Microsoft. Soporta muchas directivas y macros avanzadas.

Diferencias principales:

- NASM es open-source; MASM es propietario.
- NASM es más portable; MASM está integrado con Visual Studio.
- Sintaxis y directivas pueden variar.

Fuentes de consulta:

- Patterson, D. A., & Hennessy, J. L. *Computer Organization and Design: The Hardware/Software Interface*. Sección sobre aritmética flotante en MIPS.
- Manual de MIPS: `https://www.mips.com/products/architectures/`
- Tanenbaum, A. S. *Structured Computer Organization*.
- `https://en.wikipedia.org/wiki/Porting`
- Intel® 64 and IA-32 Architectures Software Developer Manuals.
- `https://en.wikipedia.org/wiki/X86_assembly_language`
- NASM: `https://www.nasm.us/`
- MASM: `https://docs.microsoft.com/en-us/cpp/assembler/masm/`