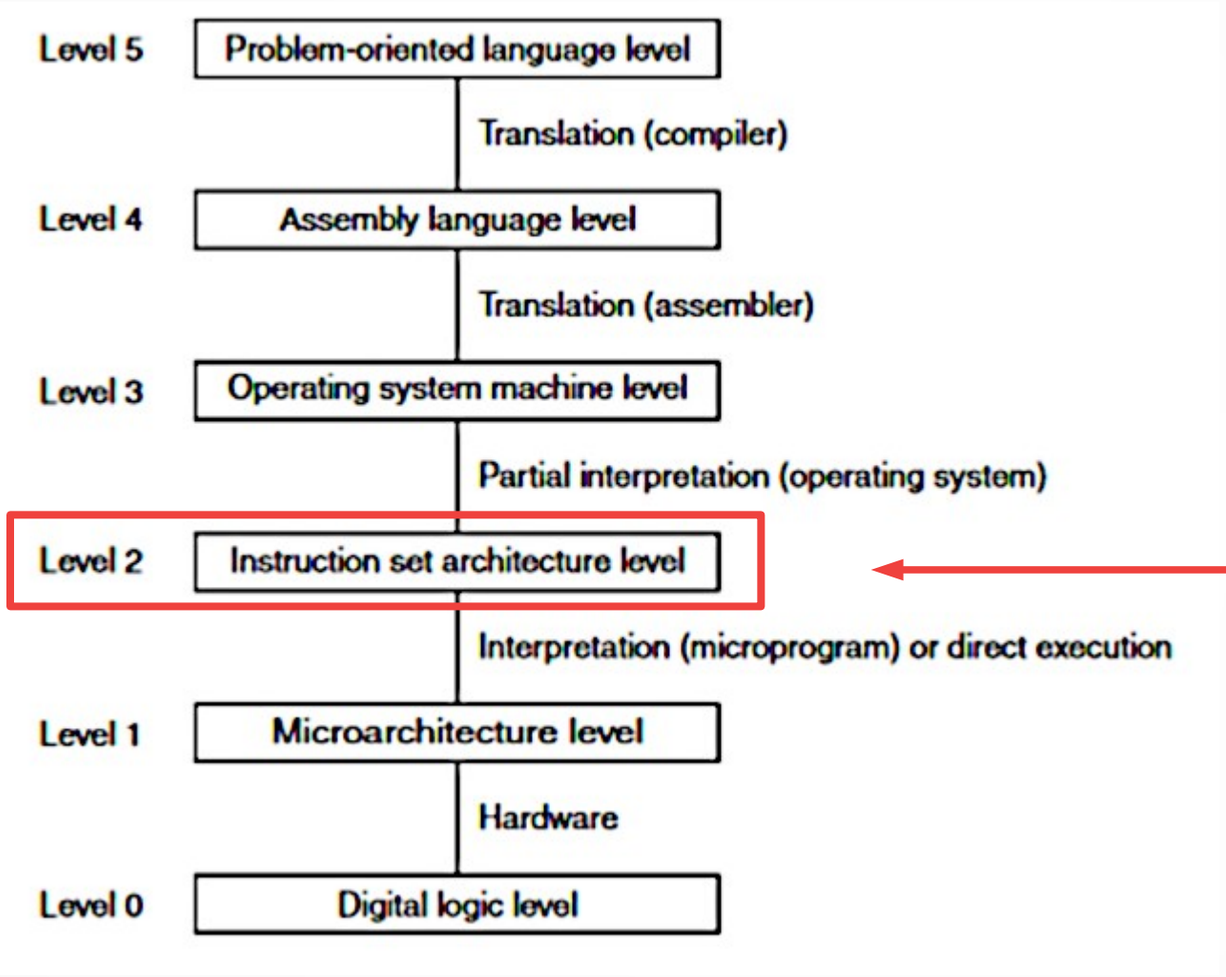
Nivel ISA – Parte 1

Arquitectura y Organización de Computadoras II

Ticiano J. Torres Peralta

REV2021



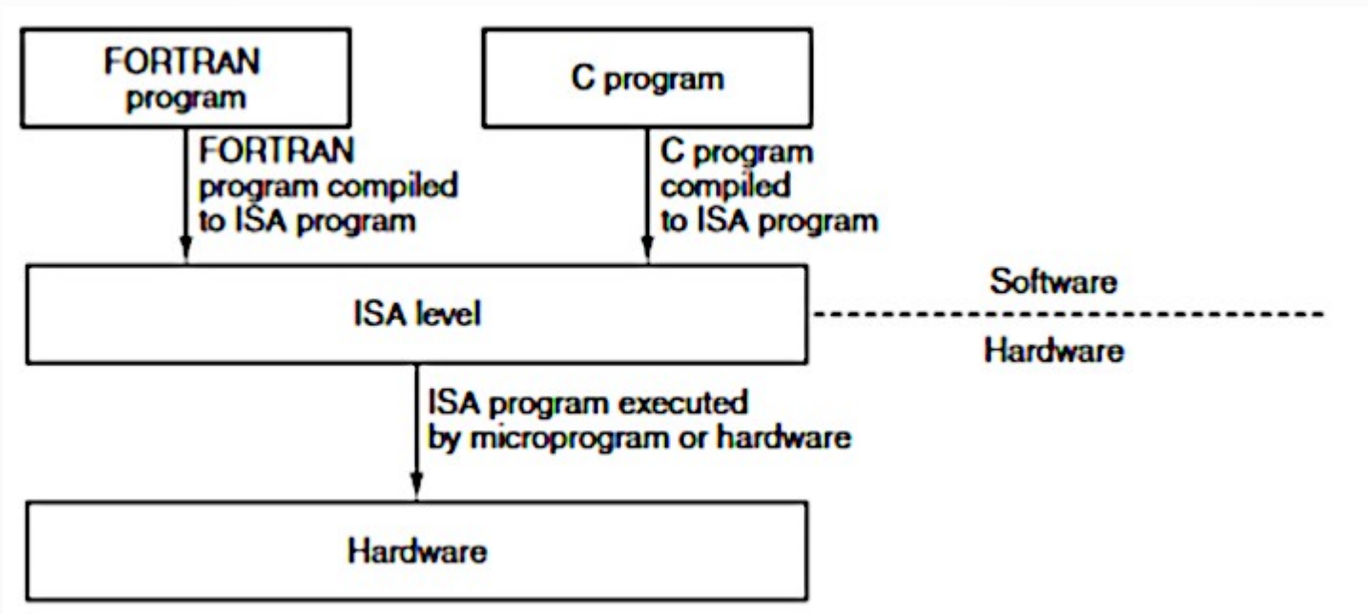
Estamos aquí

Que es el ISA

- El nivel ISA es el nivel original de la maquina multinivel contemporánea y posiblemente el mas importante de todos los niveles.
- Hoy en dia, muchos se refieren a el como la arquitectura de la computadora.
- En principio, el nivel ISA es como la machina aparece al programador que programa en lenguaje maquina.
- Mas importante, es el nivel que integra el mundo de software con el mundo de hardware.

Nivel ISA

Es el lenguaje que tanto los compiladores como el hardware tienen que entender para poder comunicase uno con el otro.



Que tan importante es el ISA?

Un arquitecto de sistema tiene que:

- Negociar con los escritores de compiladores
- Negociar con los ingenieros de hardware
- Tener en muy presente compatibilidad en reversa

Meta-propiedades de un buen ISA

Debería definir un set de instrucciones que:

1) Pueden ser implementadas en presente y futuras tecnologías, resultando en diseños que son eficientes en términos de costos, para varias generaciones de procesadores.
2) Pueden proveer una traducción clara para código compilado.
3) Todo esto mientras manteniendo compatibilidad en reversa (la realidad económica).

Quien define los ISAs

- Para muchas arquitecturas, el ISA es especificado por documentos formales, mantenidos por algún consorcio de la industria.
- Por ejemplo, el ISA para arquitecturas ARM, es publicado por ARM Ltd.
- Esto permite que fabricantes, de procesadores ARM, puedan crear sus propias implementaciones, con una variedad de precios y rendimiento, mientras garantizando la funcionalidad a nivel software.

Características de un ISA

- Como nadie es tan insano de programar a nivel maquina, podemos definir el ISA desde el punto de vista del compilador.
- Podríamos decir que el ISA es el código que el compilador entrega de salida.

Características de un ISA

Entonces, para que un escritor de compilador pueda producir este tipo de código, debe conocer:

- El modelo de memoria
- Que registros existen y están disponibles
- Que tipos de datos están disponibles
- Que instrucciones están disponibles
- *Que hardware especializado tiene la microarquitectura*

Taxonomía del nivel ISA

El factor primario que influye en el diseño de un ISA es el almacenamiento interno que tiene el procesador. Las tres elecciones principales son:

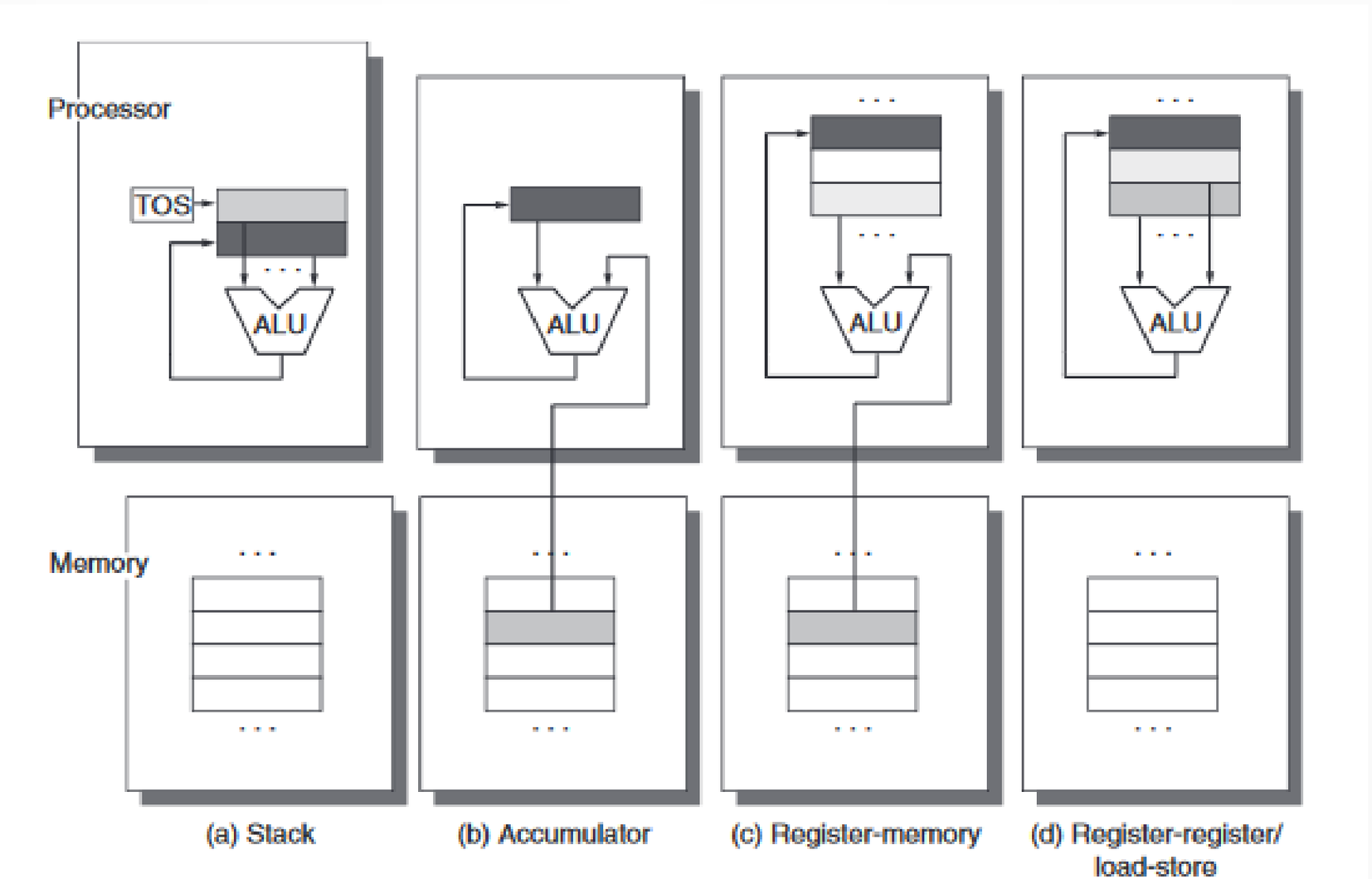
1) Una pila
2) Un acumulador
3) Un set de registros (especialmente los de propósito general)

Taxonomía del nivel ISA

La característica que afecta este factor es principalmente los operandos de la instrucciones.

1) En una arquitectura con pila, los operandos tienden a ser implícitos (existen en la pila).
2) En una arquitectura con acumulador, un operando es siempre el acumulador.
3) En una arquitectura de registros de propósito general (GPR arquitectures), los operandos son solo explícitos.

Taxonomía del nivel ISA



Ejemplo:

Una operación matemática re compilaría de manera diferente en cada tipo de arquitectura mencionada:

$$C = A + B$$

Stack	Accumulator	Register (register-memory)	Register (load-store)
Push A	Load A	Load R1,A	Load R1,A
Push B	Add B	Add R3,R1,B	Load R2,B
Add	Store C	Store R3,C	Add R3,R1,R2
Pop C			Store R3,C

Arquitecturas Modernas

- Arquitecturas modernas (después de lo 1980) son principalmente arquitecturas GPR.
- La mayoría de arquitecturas GPR usan Load-Store.
- La razones principales por la aparición de esta arquitectura son:
  - El almacenamiento interno al procesador (Registros) es MUCHO mas rápido que la memoria principal.
  - Los compiladores pueden usar a los registros de manera mas eficiente que los otros tipos de almacenamiento interno (pila, acumulador)
  - Los registros pueden ser usados para sostener variables.

Cuantos registros es suficiente?

La respuesta depende mucho en la efectividad del compilador. La mayoría de los compiladores reservan el uso de los registros para lo siguiente:

- Evaluar expresiones (matematicas/lógicas/etc.)
- Pasar parámetros
- Y los que quedan para sostener variables.

Arquitecturas GPR

- Las arquitecturas GPR se pueden dividir en tres sub tipos mas:
- Arquitecturas Registro-Memoria
- Arquitecturas Registro-Registro (Load-Store)
- Arquitecturas Memoria-Memoria (no se las encuentra hoy en día en arquitecturas modernas).

Características de arquitecturas GPR

Hay dos características importantes:

- Instrucciones del ALU pueden tener dos o tres operandos.
  1) Tres operandos: la instrucción contiene un operando para el resultado y dos operandos para la operación
  2) Dos operandos: la instrucción contiene dos operandos para la operación, y uno es re usado para guardar el resultado
- La cantidad de operandos, en la instrucción del ALU, que pueden ser direccionados desde la memoria. Puede ser desde 0 a 3.

Características de arquitecturas GPR

Number of memory addresses	Maximum number of operands allowed	Type of architecture	Examples
0	3	Load-store	Alpha, ARM, MIPS, PowerPC, SPARC, SuperH, TM32
1	2	Register-memory	IBM 360/370, Intel 80x86, Motorola 68000, TI TMS320C54x
2	2	Memory-memory	VAX (also has three-operand formats)
3	3	Memory-memory	VAX (also has two-operand formats)

Tipo	Ventaja	Desventaja
Registro-Registro (0, 3)	Simple, instrucciones de longitud fija, simple modelo de generación de código. Instrucciones toman un a cantidad similar de ciclos para ejecutar.	Los programas se compilar a un total mayor de instrucciones que los mismos en arquitecturas que permiten referencias a memoria. Mas instrucciones y una densidad menor de instrucciones causan que el programa sea mas grande.
Registro-Memoria (1, 2)	Los datos pueden ser accedidos ni primero tener que cargarlos. El formato de instrucción tiende a ser fácil de decodificar y suelen tener buena densidad.	En una operación aritmética, un operando es destruido. Codificando el numero de registro y una dirección de memoria en cada instrucción puede restringir la cantidad de registros disponibles. CPI varia dependiendo adonde esta el operando.
Memoria-Memoria (2, 2) o (3, 3)	Código mas compacto. No desperdicia registros como almacenamiento temporario.	Mucha variación en el CPI, especialmente para instrucciones con tres operandos. Mucha variación en el trabajo por instrucción. Tanto accesos a memoria causa cuellos de botella. (No se las usa hoy en día).