Representación de las Instrucciones

HISTORIAL DE REVISIONES			
NÚMERO	FECHA	MODIFICACIONES	NOMBRE
v1.0.0	2018-10-17		CA

Índice

1. Temario **1**
1.1. Bibliografía . . . 1

2. Objetivos **1**
2.1. Requisitos . . . 1

3. Introducción **1**
3.1. Diseño . . . 1
3.2. Representación . . . 2
3.3. Estructura de la Memoria . . . 2
3.3.1. Memoria Principal . . . 2

4. Elementos de una Instrucción Máquina **2**
4.1. Direcciones implícitas . . . 3
4.2. Tipos de Arquitecturas de Operando: Ejemplos . . . 3

5. Representación de las instrucciones en lenguaje ensamblador (ASM) para la arquitectura i386/amd64 **4**
5.1. Lenguaje Máquina Binario . . . 4
5.1.1. Almacenamiento en Memoria . . . 4

6. Operandos: Modos de Direccionamiento **4**
6.1. Localización . . . 4
6.2. Direcciones referenciadas durante el ciclo de instrucción . . . 5
6.3. Direccionamiento para un lenguaje general . . . 5
6.3.1. Formato de instrucción: Campos . . . 5
6.3.2. Tipos de direccionamiento . . . 6

7. Operaciones **7**
7.1. Códigos de Operación . . . 7
7.2. Tipos de Operaciones . . . 7

8. ISA de Diferentes Computadoras **8**
8.1. x86 . . . 8
8.2. Motorola M68000 . . . 8
8.3. MIPS . . . 9
8.4. ARM . . . 9

1. Temario

1. Representación de instrucciones
    a. Lenguaje máquina, lenguaje ensamblador y lenguajes de alto nivel
    b. Formato de instrucción
    c. Tipos de instrucción y modos de direccionamiento

1.1. Bibliografía

2. Objetivos

- Analizar la arquitectura del repertorio de las instrucciones máquina (Formato de instrucciones, formato de datos, operaciones y direccionamiento de operandos) de arquitecturas ISA en general.

2.1. Requisitos

- Requisitos:
    - Von Neumann Architecture: Arquitectura de una Computadora, Máquina IAS.
    - Programación en lenguaje ensamblador IAS
    - Representacion de datos
    - Operaciones Aritméticas y Lógicas

3. Introducción

3.1. Diseño

- El diseño de la arquitectura del repertorio de instrucciones (**ISA**: instruction set architecture) de una computadora comprender diseñar y definir:
    - Estructura de la Computadora: CPU-Memoria-Bus-I/O
    - La representación y formato de los datos
    - **La representación y formato de las instrucciones**
    - Repertorio de instrucciones: Las operaciones y modos de direccionamiento que ha de interpretar y ejecutar la computadora.
- La arquitectura ISA define la potencialidad de la CPU de la computadora.
- El diseño de la arquitectura ISA va a afectar al rendimiento de la computadora.
    - Programa binario resultado de la compilación del programa fuente.
    - Ocupación de Memoria
    - Implementación (dificultad) y rendimiento de la CPU.

3.2. Representación

- En este tema se trata de la representación e interpretación de las instrucciones.
- Las instrucciones se pueden representar en dos lenguajes
  - Lenguaje binario
    - El lenguaje binario implica un *formato de la instrucción*.
  - Lenguaje símbolico o lenguaje ensamblador
    - El lenguaje ensamblador implica una *sintaxis*
- La representación de las instrucciones en lenguaje binario permite su almacenamiento en la memoria principal así como facilitar el ciclo de instrucción mediante su decodificación y ejecución por parte de la CPU.
- La representación de las instrucciones en lenguaje simbólico tiene como objetivo facilitar la tarea del programador en la interpretación de las instrucciones y en el desarrollo de programas en lenguaje ensamblador.

3.3. Estructura de la Memoria

- Jerarquía de Memoria: Registros internos a la CPU y Memoria principal (DRAM) externa a la CPU

3.3.1. Memoria Principal

- Espacio de direcciones lineal: Notación hexacecimal
- Direccionamiento: bytes : notación hexadecimal

4. Elementos de una Instrucción Máquina

- Código de Operaciones:
  - La instrucción debe de especificar que operación debe de realizar la CPU. Operaciones cómo las aritméticas de suma y resta , operaciones lógicas como not y and, operaciones de transferencia de datos entre posiciones de la memoria principal, operaciones de entrada y salida como la transferencia de datos del disco duro a la memoria principal, etc
- Source Operand Reference:
  - Una operación puede requerir el procesamiento de uno o más datos. Por ejemplo la operación lógica not requiere de un operando.
- Target Operand Reference:
  - Una operación de suma requiere de dos operandos, uno es el operando fuente y otro el operando destino.
- Result Reference:
  - Una operación de suma requiere salvar el resultado de la operación.
- Next Instruction Reference:
  - Una vez finalizada la ejecución de la instrucción es necesario indicar a la CPU donde esta almacenada la próxima instrucción a ejecutar a través del Contador de Programa PC.

4.1. Direcciones implícitas

- Direcciones que no aparecen explícitamente en la instrucción. Ejemplos:
  - Próxima instrucción en el Contador de Programa
  - Resultado en el Acumulador
  - etc

4.2. Tipos de Arquitecturas de Operando: Ejemplos

- 3 Tipos
  - Arquitectura orientada a Acumulador: Un operando está implicitamente en el Acumulador
  - Arquitectura orientada a Stack ([?]):
    - Los operandos se introducen o extraen de la pila interna de la CPU
    - Concepto de pila: push/pop → empujar/extraer → el primero en entrar es el último en salir → First Input Last Output
    - SP: Registro Stack Pointer : registro que apunta al Top de la pila (parte alta de la pila)
  - Arquitectura orientada a Registros:
    - Dos tipos: Reg/Mem y Load/Store, como es el caso de la arquitectura amd64 y arm respectivamente.
    - Reg/Mem : para que la instrucción se ejecute uno de los dos operandos debe de estar en un registro
    - Load/Store: Los dos operandos deben de estar en dos registros para que dicha instrucción se ejecute
- Ejemplo: código para realizar la operación **C=A+B** en 4 arquitecturas de operando diferentes.

Stack	Acumulator	Register/Memory	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**

  - Los nombres de las variables, A, B,C son referencias a la Memoria Principal.
  - Descripción RTL
    - Stack: M[SP]←M[A],SP←SP-1; M[SP]←M[B],SP←SP-1;M[SP+1]←M[SP]+M[SP+1],SP←SP+1;
      - **Add** → NO hay referencia ni al operando fuente ni al operando destino.
      - Los operandos han de cargarse previamente en la pila
    - Acumulator: AC←M[A];AC←AC+M[B];C←M[AC]
      - **Add B** → NO hay referencia al operando DESTINO
      - El Operando destino a de cargarse previamente en el acumulador.
    - Reg/Mem: R1←M[A];R3←R1+M[B];M[C]←R3
      - **Add R3,R1,B** → NO se puede referencia a más de un operando en MEMORIA
    - Si un operando está almacenado en la memoria, el resto a de cargarse previamente en los registros.
    - Load/Store: R1←M[A];R2←M[B];R3←R1+R2;M[C]←R3.
      - **Add R3,R1,R2** →Solamente se hacen referencias a REGISTROS, ninguna referencia a memoria
      - Los operandos fuente y destino han de cargarse previamente en los registros

---

nota

La arquitectura x86 está orientada a Reg/Mem, por lo que no se puede referenciar en la misma instrucción a un operando fuente en MEMORIA y el operando destino también en MEMORIA, es decir, ambos operandos referenciados a MEMORIA.

- Ejemplo de código para realizar la opeación **(A-B)/(DxE+C)** según 4 arquitecturas ISA diferentes: arquitectura con 3 operandos referenciados, con 2 operandos referenciados, con 1 operando referenciado y ningún operando referenciado

./images/instrucciones_representacion/addresses_arch.jpg

- 4º Caso: Arquitectura de Operando tipo Stack:
  - M[SP]←M[C];M[SP]←M[E];M[SP]←M[D];MUL;ADD;M[SP]←M[B];M[SP]←M[A];SUB;DIV
  - push C; push E; push D; mul; add; push B; push A; sub; div;

5. Representación de las instrucciones en lenguaje ensamblador (ASM) para la arquitectura i386/amd64

5.1. Lenguaje Máquina Binario

5.1.1. Almacenamiento en Memoria

- Una vez realizado el proceso de traducción del módulo fuente en lenguaje ensamblador se genera un módulo objeto en lenguaje binario que se almacena en el disco duro en forma de fichero.
- El fichero que contiene el módulo objeto ejecutable en lenguaje binario es necesario cargarlo en la memoria principal. Esta tarea la realiza el **cargador** del sistema operativo.
- Cada dirección de memoria apunta a 1 byte.
- La dirección más baja apunta a todo el objeto: instrucción o dato.
- Ejemplo:
  - **4001a4: 48 83 ec 10**
    - En la posición **0x4001A4** está el byte **48**
    - En la posición **0x4001A4+1** está el byte **83**
    - En la posición **0x4001A4+2** está el byte **EC**
    - En la posición **0x4001A4+3** está el byte **10**
    - En la posición de memoria principal `0x4001A4` está almacenada la instrucción de 4 Bytes

6. Operandos: Modos de Direccionamiento

6.1. Localización

- Posibles ubicaciones de los operandos.
  - En la propia instrucción
  - Memoria interna: registros CPU
  - Memoria Principal: memoria DRAM
  - Memoria i/o: registros en controladores de entrada/salidas denominados puertos.

6.2. Direcciones referenciadas durante el ciclo de instrucción

- Durante el ciclo de instrucción se pueden referenciar:
  - Una dirección para referenciar a la instrucción
  - Una dirección para el operando primero
  - Una dirección para el operando segundo
  - Una dirección para el resultado
  - Una dirección que referencie a la siguiente instrucción
- Tipos de instrucciones según el número de direcciones referenciadas durante su ejecución.
  - Instrucciones sin operando, con un operando, con múltiples operandos.
  - Depende de la arquitectura: Acumulador (Ej: máquina IAS), Registro-Memoria(Ej: máquina x86), Load/Store (Ej:ARM), Stack (Ej: máquina JVM), Memoria-Memoria
  - referencias implicitas al operando

6.3. Direccionamiento para un lenguaje general

6.3.1. Formato de instrucción: Campos

- **Ejemplo particular** de una estructura del formato de instrucción en tres campos en una arquitectura ISA.
  - Código de Operación: mover, cargar, sumar, restar, etc
  - Código A: campo de operando : hace referencia a la localización del operando
  - Código Mod. Direc: representa el modo de interpretar el campo A
- EA: Efective Address : Dirección efectiva donde está localizado el operando
- Op: Operando .Es el dato contenido en la dirección efectiva EA.
- Los datos *operando* Op pueden estar almacenados en:
  1. Memoria externa RAM
     a. Una dirección de memoria conteniendo un dato.
     b. Una dirección de memoria conteniendo una instrucción. El dato es uno de los campos de a propia instrucción. Direccionamieno Inmediato.
  2. Memoria interna GPR
     a. Registros rax,eax,...

6.3.2. Tipos de direccionamiento

- La *dirección* de referencia efectiva E.A. de la ubicación del operando se obtiene según los distintos modos de direccionamiento.
- El modo de direccionamiento está codificado en el campo M.D.
- Inmediato:
  - El operando se obtiene del campo de la propia instrucción.
  - EA= no existe
  - Op=A
- Directo:
  - El operando está en la memoria externa. El campo de operando contiene la dirección efectiva
  - EA=A
  - Op=M[EA]
- Registro:
  - El operando está en la memoria interna. El campo de operando contiene la referencia del Registro.
  - EA=A
  - Op=R
- Indirecto:
  - La dirección efectiva esta almacenada en una posición de memoria externa o interna.
  - EA=M[A] o R
  - Op=M[M[A]] o M[R]
- Desplazamiento:
  - La dirección efectiva del operando se obtiene mediante una operación aritmética entre una dirección base y un desplazamiento relativo a la dirección base. La dirección base se toma como referencia y el desplazamiento es relativo a la dirección base.
    a. Relativo al contador de programa PC:
       - La dirección base es implíctamente el contador de programa y el desplazamiento está en el campo de operando.
       - EA=PC+A
       - Op=M[EA]

    b. Relativo a Base:
       - El desplazamiento está en el campo de operando y la dirección base está en el registro.
       - EA=R+A
       - Op=M[EA]

    c. Indexado:
       - El desplazamiento está en el registro y la dirección base está en el campo de operando.
       - EA=A+R
       - Op=M[EA]
- Para hacer referencia a los operandos fuente o destino la arquitectura de la instrucción es muy *flexible* ya que se dispone de distintos modos de direccionar dichos operandos.

7. Operaciones

7.1. Códigos de Operación

- La codificación del conjunto de operaciones depende de cada arquitectura ISA.

7.2. Tipos de Operaciones

- Categorías:
  - Data Processing: Arithmetic and logic instructions
  - Data Load/Store: Movement of data into or out of register and/or memory locations
  - Data Movement: I/O instructions
  - Control: Test and Branch instructions

tipos de instrucciones

- El repertorio puede ser: reducido/extenso, complejo/sencillo.

8. ISA de Diferentes Computadoras

8.1. x86

- Felix Cloutier

8.2. Motorola M68000

- Referencias
  - Instruction Set Basic
  - Wikibook
  - Manual de Referencia
- arquitectura general

```
2 versiones: Procesador de 16 bits ó 32 bits
Aprox . 90 instrucciones máquina
12 modos de  direccionamiento
9 formatos de instrucción distintos y con tamaños de una a cinco palabras
Ancho del  bus  de datos: 16 bits ó 32 bits
Tamaño mínimo  direccionable : 1 byte
Ancho del  bus  de direcciones: 24 bits (2^24^bytes = 16 Mbytes de memoria direccionables  ↩
    )
```

- Registros:
  - 8 Registros de Datos de propósito general (16/32): D0-D7
  - 7 Registros de Instrucciones de propósito general (16/32) :A0-A6
- modos de direccionamiento
  - # : inmediato
  - Di : registro directo
  - (Ai): indirecto de registro
  - +(Ai): indirecto de registro con postincremento con la escala del tamaño del operando (1,2,4 bytes)
  - (Ai)+: indirecto de registro con postincremento con la escala del tamaño del operando
  - -(Ai): indirecto de registro con predecremento con la escala del tamaño del operando
  - (Ai)-: indirecto de registro con preincremento con la escala del tamaño del operando
  - D(Ai): indirecto de registro con desplazamiento D
  - D(Ai,Ri.X) : registro Ai indirecto indexado Ri con desplazamiento D
  - D(PC) : relativo al PC con desplazamiento D
  - D(PC,Ri.X) : relativo al PC indexado Ri con desplazamiento D
- Datos
  - Enteros en Complemento a 2 .
  - Sufijos Operación: B byte (1 byte), W word (2 bytes) , L long (4 Byte)
  - Prejijos datos: $ hexadecimal
- Memoria
  - Big Endian : LSB en la dirección más alta y MSB en la dirección más baja

8.3. MIPS

- Procesador con una arquitectura de 32 bits
- Microprocessor without Interlocked Pipeline Stages (MIPS) Architecture
- Versiones de la arquitectura MIPS:
  - MIPS I (R2000 cpu), II (R6000), III (R4000), IV (R8000, R5000, R10000), and V (nunca implementada);
  - MIPS32/64 :MIPS32 is based on MIPS II with some additional features from MIPS III, MIPS IV, and MIPS V; MIPS64 is based on MIPS V

```
70 instrucciones máquina
Instrucciones clasificadas en cuatro grupos
        Movimiento de datos
Aritmética entera, logicas y desplazamiento
        Control de flujo
        Aritmética en punto flotante
4 modos de  direccionamiento
        Inmediato
        Directo de registros
        Indirecto con desplazamiento
        Indirecto con desplazamiento relativo al PC
Banco de 64 registros (32 bits cada uno)
        32 de propósito general (R0-R31)
        32 para instrucciones en punto flotante (F0-F31). Pueden usarse como:
        32 registros para operaciones en simple precisión (32 bits)
        16 registros para operaciones en doble precisión (64 bit)
3 formatos de instrucción distintos con longitud única de 32 bits:
         Op Code: 6 bits
         R :three registers, a shift amount field, and a function field;
         I :two registers and a 16-bit immediate value
         J :26-bit jump target
Arquitectura registro-registro
        Sólo las instrucciones de LOAD y STORE hacen referencia a memoria
        El resto de instrucciones operan sobre registros
                Instrucciones con tres operandos: 2 op.fuente y 1 op.Destino

Notación ensamblador: op  x, y, z    x<-(y)op(z)
Datos:
        Enteros Complemento a 2 : byte (1B), media palabra (2B), palabra (4B)
        Nº Reales: IEEE-754 simple y doble precisión
```

- MIPS architecture
- Versiones de la ISA MIPS
- procesadores con arquitectura MIPS: R2000, etc
- quick tutorial
- Emulador MIPS Online

8.4. ARM

- x