Representación de instrucciones

Organización de Computadoras 2019

Universidad Nacional de Quilmes

1. Ciclo de vida de un programa

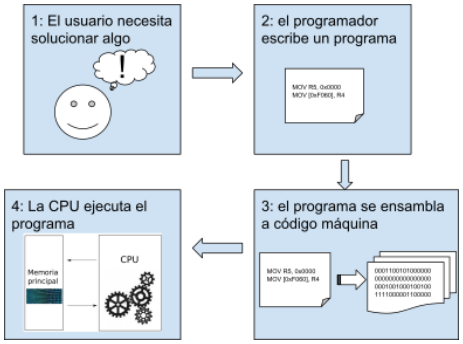
Las computadoras funcionan siguiendo una lista de instrucciones que se les dan, y que les permiten ordenar, encontrar y generar información. Pero estas instrucciones deben estar disponibles de alguna manera, en alguna codificación, que la computadora pueda interpretar y traducir en las acciones que se requieren para cada instrucción. Las computadoras actuales están construidas con circuitos digitales y por lo tanto necesitan que la mencionada codificación se le provea como cadenas binarias.

Por ejemplo, para escribir el programa que resuelva `A=A-B` se necesita poder escribir la instrucción: `restar LO-QUE-HAY-EN-B a LO-QUE-HAY-EN-A`. Entonces surge la necesidad de tener una representación binaria, que pueda ser entendida por la unidad de control, quien le dirá a la ALU que debe activar una resta, y por lo tanto que debe usar el circuito restador. Esa representación se denomina **código máquina** y debe incluir en cada instrucción la siguiente información:

- Qué operación es (suma, resta,etc)
- Cuáles son los operandos
- Dónde se guarda el resultado

En este punto, es razonable hacerse la pregunta: ¿Qué son A y B? Estas son variables (como las de matemática) que almacenan cada una un valor. Para tal fin, en una computadora los valores de las variables deben ser mantenidos en dispositivos que se denominan **Registros**. En este caso necesitaremos un registro para A y otro para B. El resultado se almacena en el registro A: `A=A-B`.

A continuación se muestran los pasos del ciclo de vida de un programa:



1.1. Elaboración de un programa

Los detalles que se explican en esta sección pueden no parecer importantes cuando se está programando con lenguajes de alto nivel como Gobstones o Java donde las características de la arquitectura no son tan *visibles*, pero es necesario conocer cómo opera la computadora internamente para administrar bien los recursos al momento de programar.

Un punto de encuentro en que el/la diseñador/a del computador y el/la programador/a pueden ver la misma máquina es el **repertorio de instrucciones**. Desde el punto de vista de la persona que diseña, el conjunto de instrucciones máquina constituye la especificación o requisitos funcionales del procesador: implementar el procesador es una tarea que, en buena parte, implica implementar el repertorio de instrucciones máquina. Desde el punto de vista de la persona que programa, las instrucciones de ese lenguaje ensamblador son las herramientas disponibles para expresar los programas. Además, debe conocer la

estructura de registros y de memoria, los tipos de datos que acepta directamente la máquina y el funcionamiento (o restricciones) de la ALU (Unidad Aritmético-lógica).

El funcionamiento del procesador está determinado por las instrucciones que ejecuta. Estas instrucciones se denominan **instrucciones máquina** o instrucciones del computador. Al conjunto de instrucciones distintas que puede ejecutar el procesador se denomina **repertorio de instrucciones**.

Cada instrucción debe contener la información que necesita el procesador para su ejecución. Dichos elementos son:

- **Código de operación:** especifica la operación a realizar (suma, resta, etc). La operación se indica mediante un código binario denominado código de operación, o de manera abreviada, **codop**.
- **Referencia a los operandos origen:** la operación puede implicar uno o más operandos origen, es decir operandos que son entradas para la operación.
- **Referencia al resultado:** la operación puede producir un resultado por lo que puede ser necesario indicar donde se almacenará.
- **Referencia a la siguiente instrucción:** indica al procesador de dónde captar la siguiente instrucción tras completarse la ejecución de la instrucción actual

La siguiente instrucción a captar está en memoria principal o, en el caso de un sistema de memoria virtual en memoria principal o en memoria secundaria (disco). En la mayoría de los casos, la siguiente instrucción a captar sigue inmediatamente a la instrucción en ejecución y en tales casos no hay referencia explícita a la siguiente instrucción. Cuando sea necesaria la referencia explícita debe suministrarse la dirección de memoria principal o de memoria virtual.

Los operandos y el resultado pueden estar en alguna de las tres áreas:

- **Memoria principal o virtual**: como en las referencias a instrucciones siguientes, debe indicarse la dirección de memoria principal o memoria virtual.
- **Registro del procesador**: Salvo raras excepciones, un procesador contiene uno o más registros que pueden ser referenciados por instrucciones máquina. Si sólo existe un registro, la referencia a él puede ser implícita. Si existe más de uno, cada registro tendrá asignado un número único y la instrucción debe contener el número del registro deseado.
- **Dispositivo de entrada/salida**:la instrucción debe especificar el módulo y dispositivo de Entrada/Salida para la operación. En el caso de E/S asignadas en memoria, se dará otra dirección de memoria principal o virtual.

1.2. Ensamblado del programa

Originalmente el/la programador/a escribía sus programas en código máquina, pero es difícil para los humanos manejar las **representaciones binarias** y hace que la programación sea una tarea muy engorrosa y propensa a errores. Por ello, la evolución de la programación propuso utilizar **representaciones simbólicas** para las instrucciones máquina, donde los **codops** se representan mediante abreviaturas, denominadas *nemotécnicos*, que indican la operación en cuestión. Ejemplos usuales son:

ADD	Sumar
SUB	Restar
MUL	Multiplicar
DIV	Dividir
LOAD	Cargar datos desde memoria
STORE	Almacenar datos en memoria

Los operandos también suelen representarse simbólicamente. Por ejemplo, la instrucción **ADD R,Y** puede significar sumar el valor contenido en la posición de datos **"Y"** al contenido del registro **R"**. En este ejemplo, **Y** hace referencia a la dirección de una posición de memoria, y **R** a un registro particular. Observe que la operación se realiza con el contenido de la posición y no con su dirección.

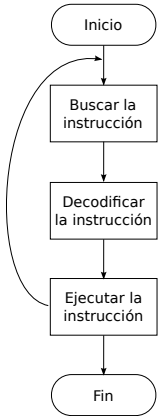
Pero esta representación requirió el desarrollo de un dispositivo que tradujera dicho código simbólico al lenguaje de la computadora. Este lenguaje simpólico se denomina **código fuente**, y el mencionado dispositivo es el **ensamblador**,

quien además, carga el programa en memoria principal. Al proceso de traducir del código fuente a código máquina se lo denomina **Ensamblar**.

Es raro encontrar ya programadores en lenguaje máquina. La mayoría de los programas actuales se escriben en un lenguaje de alto nivel o, en ausencia del mismo, en lenguaje ensamblador.

1.3. Ejecución de un programa

En particular, la ejecución de un programa es la ejecución de cada instrucción, dando lugar al siguiente ciclo de ejecución de instrucción que se muestra en la siguiente figura:



Ciclo de ejecución de una instrucción

Durante su ejecución, la instrucción se escribe en un **registro de instrucción (IR)** del procesador y éste debe ser capaz de extraer los datos de los distintos campos de la instrucción para realizar la operación requerida.

Veamos un ejemplo con una instrucción ADD:

- Al comienzo del ciclo (de ejecución del programa, no confundir con el ciclo de vida de un programa) la UC hace la lectura de instrucción, obteniendo la cadena correspondiente.
- La UC decodifica la instrucción, identificando un ADD, que se traduce en una señal a la ALU pidiendo una suma, y una copia de los operandos a las entradas de la ALU. En Q1, el operando destino puede ser uno de los 8 registros (R0 a R7), y el operando origen puede ser además un valor inmediato (que está incluido en la instrucción).
- La ALU ejecuta la suma y el resultado se redirige al registro R0 (a través de un demultiplexor).

2. Arquitectura Q1

Para poner en práctica los conceptos utilizaremos una arquitectura conceptual que llamaremos Q. La misma está pensada en versiones acumulativas y la primera recibe el nombre de **Q1**, con la cual relacionaremos las diferentes etapas que intervienen en el ciclo de vida de un programa, y del ciclo de ejecución de las instrucciones.

Q1 tiene las siguientes características:

- 8 registros de 16 bits visibles al programador, que son variables disponibles para usar en los programas, denominados `R0` a `R7`.
- 5 instrucciones de 2 operandos: MUL, ADD, DIV, SUB y MOV
- Los operandos pueden ser variables (uno de los registros) o constantes.
- Los valores constantes tienen 16 bits, y se escriben en hexadecimal con la sintaxis: `0xAAAA`
- Al primer operando se lo denomina **operando destino**, por ser además donde se almacenará el resultado, y al segundo se lo denomina **operando origen**.

Por ejemplo, la instrucción `MOV R0, R1` copia el valor del registro R1 al registro R0. Por otro lado, la instrucción `ADD R5, 0x05AB` suma el valor `05AB` al contenido del registro R5.

Ensamblar instrucciones en Q1

Para que los programas escritos en el lenguaje de Q1 sean ejecutables por una computadora, deben estar escritos en código máquina. Entonces es necesario establecer las reglas de traducción código fuente a código máquina que debe aplicar un ensamblador. Esas reglas se conocen como **formatos de instrucciones**. Estos definen la

organización de los bits dentro de una instrucción en términos de las partes que la componen. La instrucción debe incluir información acerca de **la operación a ejecutar** y sobre **qué datos**. Entonces, mínimamente debe incluir el código de la operación y un mecanismo para llegar hasta el/los operando/s, mediante lo que llamamos **modos de direccionamientos**. La siguiente figura muestra un ejemplo del formato de una instrucción de dos operandos en Q1:

CodOp (4b)	Modo Dest (6b)	Modo Origen (6b)	Destino (16b)	Origen (16b)

Cada uno de estos

El campo de **codop** se completa siguiendo la tabla que sigue, donde se muestran los códigos de operación para cada operación de dos operandos, que son las disponibles en la versión **Q1**:

Operación	Cod Op
MUL	0000
MOV	0001
ADD	0010
SUB	0011
DIV	0111

Existen varios mecanismos para indicar dónde están los operandos, y estos reciben el nombre de **modos de direccionamiento**. En el caso de nuestra Arquitectura Q1, veremos por ahora dos, **modo registro** y **modo inmediato.**

Los campos **Modo Destino** y **Modo Origen** responden a la siguiente codificación de modos de direccionamiento:

Modo	Codificación
Inmediato	`000000`
Registro	`100rrr`

Donde `rrr` es una codificación (en 3 bits) del número de registro.

Entonces, las instrucciones arriba mencionadas se ensamblan como se muestra a continuación (Notar que el codigo maquina se expresa en binario y luego se compacta en hexadecimal a los fines de una mejor legibilidad):

Código fuente	Código máquina	Hexa
`MOV R0, R1`	`0001 100000 100001`	`1881`
`ADD R5, 0x05AB`	`0010 100101 000000` `0000 0101 1010 1011`	`294005AB`

Fuente

Extraído de *Organización y Arquitectura de Computadores*, William Stallings - 7ta edición. Capítulo 10mo.