Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen X Edición No X – Año XXXX

Visión moderna de los cálculos básicos en una computadora con estructura de Jhon Von Neumann.

Villarreal R., Pablo A.

Universidad Nacional Abierta (UNA) Carrera Ingeniería de Sistemas y **Universidad Valle del Momboy** (UVM) Facultad de Ingeniería, Escuela Ingeniería de Computación.
pvillarreal@una.edu.ve, villarrealp@uvm.edu.ve

Recibido 28/04/2017 Aceptado 30 /05/2017

Resumen

El siguiente articulo tiene como premisa el de motivar a los profesores y estudiantes en el estudio de contenidos sobre Organización de Computadoras, partiendo de la base del conocimiento básico sobre arquitectura y tecnología de computadores. Se espera que la temática abordaba en esta investigación pueda mejorar las prestaciones en calidad de servicio sobre la temática seleccionada, por ende en el afinamiento y resguardo del conocimiento científico de estudiantes de Ingeniería y del profesorado en la facultad de la Universidad Valle del Momboy y del área de Ingeniería carrera Ingeniería de Sistemas de la Universidad Nacional Abierta. El objetivo principal del artículo es el de presentar los tipos de instrucciones utilizados para realizar cálculos sencillos en una máquina computacional con arquitectura Von Neumann. Para ello expondré las consideraciones básicas de las estructuras computacionales existentes o también conocidas como estrategias, entre ellas: CISC[1] y RISC[2], para concluir se expondrá una concreción sobre un debate entre estas dos estrategias bajo unas premisas o interrogantes motivadoras ¿es mejor completar una actividad de gran tamaño ejecutando muchas tareas pequeñas pero sencillas o ejecutando unas tareas de tamaño mediano?
Palabras claves: arquitectura de computadoras, cálculos sencillos, registros, direcciones.

Abstract

The following article has the premise of motivating teachers and students in the study of content on Computer Organization, based on the basic knowledge of computer architecture and technology. It is hoped that the thematic approach in this research could improve the service quality on the selected theme, therefore in the tuning and safeguard of the scientific knowledge of Engineering students and the faculty at the

[1] CISC se conoce como el computador con un conjunto complejo o ampliado de instrucciones máquinas (complex instruction set computer).
[2] RISC acrónimo de reduced instruction set computer o computador con un conjunto reducido de instrucciones máquinas.

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen X Edición No X – Año XXXX

University of Valle del Momboy and the Area of Engineering, systems engineer of the National Open University. The main objective of the article is to present the types of instructions used to perform simple calculations in a computational machine with Von Neumann architecture. In order to do this I will present the basic considerations of the existing computational structures or also known as strategies, among them: CISC and RISC, in order to conclude a statement about a debate between these two strategies under premises or motivating questions, is it better to complete an activity of Large size by running many small but simple tasks or running a medium size task?
Keywords: computer architecture, simple calculations, records, addresses.

Introducción

Las arquitecturas de computadoras abren una brecha para los diseñadores de tecnologías computacionales, teniendo en cuenta la forma como deben perfilar su estructura de funcionamiento de cálculo, particularmente los cálculos básicos realizados en una máquina de cómputo bajo las premisas sobre una arquitectura Von Neuman. En la actualidad la computación se rige por una arquitectura de computadores donde resulta menester analizar las dos estrategias más prominentes para el diseño de computadores, estas son: RISC (reduced instruction set computer) y CISC (complex instruction set computer). El término reducido se refiere a la complejidad de las instrucciones individuales, pues la mayoría de estas instrucciones se llevan a cabo en sola operación bien definida (sumar dos valores y almacenar el resultado). Por el contrario, el término complejo puede realizar dos operaciones (sumar dos valores, almacenar el resultado e incrementar uno o dos registros).(Kane, Heinrich, 1992)

En la historia el método CISC fue creado como apoyo al programador en lenguaje ensamblador. Al respecto, con una instrucción CISC puede realizar múltiples operaciones los programadores de bajo nivel escriben pocas instrucciones para llevar a cabo una tarea de gran magnitud. Por el contrario, en el método RISC la programación se realiza en lenguajes de alto nivel, como C o Pascal. Bajo este panorama, existen pocos incentivos para reducir el número de instrucciones que necesita escribir el

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen X Edición No X – Año XXXX

programador, y el énfasis se torna alrededor en la mejora de la eficiencia de cada instrucción máquina.

Pareciera una simplicidad, pero se puede caer en la tentación de abstracción temática, cuando en realidad se deriva en un debate que debemos abordar en los espacios universitarios donde se imparte la ingeniería de sistemas, computación u otra área afín.

Las estrategias RISC y CISC se reducen a una interrogante básica:

¿Es mejor completar una tarea de gran tamaño ejecutando muchas tareas pequeñas pero sencillas (RISC) o efectuando unas cuantas tareas de tamaño mediano (CISC)?

Pareciera que la división de tareas es un asunto arbitrario. Cuando con la estrategia CISC se reduce el número de instrucciones aumentando la complejidad del procesador, mientras que con la estrategia RISC aumenta el número de instrucciones, simplificando la estructura del procesador.

Los resultados sobre estas estrategias están presentes en el mercado de los procesadores se han construidos más procesadores RISC que CISC, dos empresas se han ocupado de estos escenarios por un lado INTEL (RISC) y AMD (CISC), de repente según sus gustos sobre las arquitecturas Usted ha colocado en tela de juicio algún procesador de estas dos empresas, desde su calentamiento hasta la velocidad de computo.

El escenario planteado con anterioridad es muy somero se ha colocado para hacer un simple ejercicio de asimilación de su vivencia y la relación procesador-usuario.

He tratado de introducir este escenario de las estrategias de diseño para proseguir la de los cálculos, para ello es necesario mencionar los tipos básicos de operaciones: operaciones de transferencia de datos, operaciones de manipulación de datos y operaciones de flujo de control.

En la cual las operaciones de transferencia de datos se usan para transferir valores entre los componentes (Unidad de Memoria y Unidad Central de Proceso) de un sistema de cómputo. Mientras que las operaciones de manipulación de datos se usan

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen X Edición No X – Año XXXX

para calcular nuevos valores de datos (por ejemplo: suma y multiplicación) y las operaciones de flujo de control sirven para controlar qué operaciones se llevan a cabo.

El presente artículo está estructurado de la siguiente forma: se disertará sobre los tipos de instrucciones utilizadas para las operaciones de manipulación datos teniendo en cuenta el número de direcciones en una instrucción, también se abordaran algunas operaciones de transferencia de datos.

Desarrollo

En alguna ocasión como investigadores sobre este apasionado mundo de la computación nos hemos encontrado en esta disyuntiva ¿Cuántas direcciones? Específicamente las direcciones las tomaremos en consideración para referenciar a cálculos básicos en los cuales tratare de exponer la problemática que se presenta por considerarse un tema de estudio riguroso y de un entramado difícil de abordar. Iniciemos, suponga o considere la siguiente expresión declarativa:

$$x_1 = x_1 * x_2 + x_3 * x_4 * x_5;$$

Si observamos con cautela esta expresión se nos viene a la mente de forma inmediata las reglas de precedencia y asociatividad o el orden de prioridad de los operadores aritméticos, por formar parte del conocimiento común. Sin embargo, en los lenguajes de programación (C++, Pascal) tienen incorporadas estas reglas, pero no es muy razonable que una máquina pueda aplicar estas reglas de forma inmediata, pues su naturaleza no lo permite, y es acá donde interviene la razón humana, expresada en la mano del programador.

Para simplificar la interpretación de nuestros programas, se debe comenzar por establecer el requisito de cada enunciado especifique la aplicación de una sola operación. Así se puede simplificar esta interpretación al eliminar la necesidad de asociatividad y la precedencia de los operadores, siendo una regla opcional la secuencia, que nos ayudará para identificar la siguiente instrucción. Algo difícil de comprender, pero bajo una simplicidad manejable, revisemos el siguiente caso.

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen X Edición No X – Año XXXX

Caso 1. Revisemos la expresión $x_1 = x_1 * x_2 + x_3 * x_4 * x_5$, al utilizar una operación por enunciado, se tiene:

$$\mathbf{x_1 = x_1 * x_2;\ t_1 = x_3 * x_4;\ t_1 = t_1 * x_5;\ x_1 = x_1 + t_1;}$$

En el caso anterior podemos deducir que hace cada operador y su funcionamiento de la secuencia, de esta forma se comprenderá el fragmento presentado anteriormente, se ha introducido una variable nueva llamada t_1, en el enunciado del caso presentado a este tipo de variables se les conoce como variables temporales, pues almacenan información intermedia mientras se calculan otros resultados.

En este caso se necesitan tres resultados intermedios a saber:

Variable	Expresión	Declaración
x_1	$x_1 = x_1 * x_2$	Variable a usada para almacenar el primer resultado intermedio.
t_1	$t_1 = x_3 * x_4$	Variable usada para almacenar el segundo resultado intermedio.
t_1	$t_1 = t_1 * x_5$	Variable usada para almacenar el tercer valor dado que el programa no necesita el valor de x_2.

O básicamente: $x_1 * x_2$, $x_3 * x_4$, $t_1 * x_5$. Se conoce que el tiempo de procesamiento en un computador es de suma importancia, en tal sentido lector puede inferir que a la expresión mostrada pueda ser más reducida o mejorarla en la sintaxis, a este pensamiento se le conoce como optimización[3] de código.

Si nos planteamos esta interrogante ¿Cuál es la operación en el primer enunciado del caso?, de forma inmediata responderíamos que es la multiplicación. Sin embargo, esta respuesta no es la más idónea desde una perspectiva técnica puesto que el primer enunciado requiere además de la multiplicación su almacenamiento, o también multiplicación y almacenamiento como operación ternaria con dos operandos para la multiplicación y uno específico para almacenar el resultado llamad también operando destino, siendo los operandos los que contienen los valores en la operación binaria conocidos como operandos fuentes.

Las dos operaciones usadas en el Caso 1, multiplicación y almacenamiento y suma y almacenamiento, se consideran ternarias. Otras operaciones binarias destacan

[3] La expresión optimización no significa la mejor, en virtud de que no se puede afirmar que es la mejor solución posible, sino una mejor. Es decir, se acoge la definición optimización como la mejora.

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen X Edición No X – Año XXXX

en el caso, copia (a = b) y negación (a = -b), el caso planteado se hace como finalidad instruccional para ejemplificar como a través de las máquinas secuenciales se añaden un nuevo operando a cada operación y este es el destino. En tal sentido, la inserción de este operando las operaciones se convierten en operaciones binarias y las binarias en ternarias.

Para vislumbrar esta forma de trabajo computacional se detallará en lo inmediato cuatro estrategias para proporcionar cálculos básicos en los cuales las operaciones aritméticas comunes, entre ellas: máquinas de tres direcciones, máquinas de dos direcciones máquinas de una dirección y máquinas de cero direcciones.

Como mencione anteriormente la intencionalidad del artículo es ahondar sobre cómo se realizan estos cálculos la taxonomía presentada esta en función de un accionar academicista para ejemplificar los cálculos, porque pocas máquinas caen con claridad en algunas de estas taxonomías. Para efectos de practicidad cada máquina se expondrá bajo tres parámetros a efectos de ejemplificar estas situaciones, en tal sentido se abordará en una tabla las instrucciones y el convenio de codificación y un caso.

Máquinas de tres direcciones.

En una máquina de tres direcciones los tres operandos están explícitos en cada instrucción, a través de la siguiente tabla se presentan las instrucciones y el convenio de codificación.

Tabla 1. Instrucciones para una máquina de tres direcciones

Instrucción	Detalle
SUMAR dest, fte_1, fte_2	Sumar el valor almacenado en la posición de memoria fte_1 al valor en la posición de memoria fte_2 y almacenar el resultado en la posición de memoria dest. $M[dest] = M[fte_1] + M[fte_2]$
MULTI dest, fte_1, fte_2	Multiplicar el valor almacenado en la posición de memoria fte_1 al valor en la posición de memoria fte_2 y almacenar el resultado en la posición de memoria dest. $M[dest] = M[fte_1] * M[fte_2]$
Convenio de codificación: Primero el destino. Puede observarse que se ha listado el destino como el primer operando de cada instrucción, algunas arquitecturas de computadores listan el destino como último operando.	

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen X Edición No X – Año XXXX

}Caso 2. Usando las instrucciones de la tabla 1 se evalua la expresión siguiente:

$$x_1 = x_1 * x_2 + x_3 * x_4 * x_5;$$

Supongamos que las variables x_1, x_2, x_3, x_4 y x_5 están asociadas a las direcciones de memoria 64x, 68x, 6Cx, 70x, y 74x. Además, la existencia de la dirección de memoria C8x para usarla temporalmente.

Una posible solución al caso 2 usando las instrucciones de la tabla 1.

Operaciones	Significado didáctico	Convenio de codificación

MULTI 64x, 64x, 68x
MULTI C8x, 6Cx, 70x
MULTI C8x, C8x, 70x
SUMAR 64x, 64x, C8x
	$x_1 = x_1 * x_2$
	$t_1 = x_3 * x_4$
	$t_1 = t_1 * x_5$
	$x_1 = x_1 + t_1$
		Cada linea de las operaciones es un código y cada enunciado esta en una linea de código separada, existe entonces una relación muy estrecha entre las lineas y los enunciados. Es relación es un requisito en los ensambladores, esto se le conoce también como *orientación a lineas*.

Máquinas de dos direcciones.

Si revisamos al detalle el Caso 2 se ocurre que tres de las cuatro instrucciones repiten una dirección de memoria y que la segunda instrucción usa tres direcciones diferentes, es una frecuencia ocurrente en esta situación. En una máquina de dos direcciones, las instrucciones solo tienen dos direcciones explicitas, una de las cuales se asocia la función de servir como dirección de un operando fuente y como dirección del destino.

Tabla 2. Instrucciones para una máquina de dos direcciones

Instrucción	**Detalle**
SUMAR dest, dir	Sumar el valor almacenado en la posición de memoria fte al valor en la posición de memoria dest y almacenar el resultado en la posición de memoria dest. M[dest] = M[dest] + M[fte]
MULTI dest, dir	Multiplicar el valor almacenado en la posición de memoria fte al valor en la posición de memoria dest y almacenar el resultado en la posición de memoria dest. M[dest] = M[dest] * M[fte]
MOVER dest, dir	Mover el valor almacenado en la posición de memoria fte a la posición de memoria dest. M[dest] = M[fte]

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen X Edición No X – Año XXXX

Convenio de codificación: se muestra en estas instrucciones que la primera dirección tiene una función binaria, ellas son; el de indicar un operando fuente y el operador destino en las operaciones SUMA y MULTI, estas operaciones se le conoce también como operaciones de manipulación de datos y MOVER como una operación de transferencia de datos

Caso 3. Usando las instrucciones de la tabla 2 se evalua la expresión siguiente:

$$x_1 = x_1 * x_2 + x_3 * x_4 * x_5;$$

Supongamos que las variables x_1, x_2, x_3, x_4 y x_5 están asociadas a las direcciones de memoria 64x, 68x, 6Cx, 70x, y 74x. Además, la existencia de la dirección de memoria C8x para usarla temporalmente.

Una posible solución al caso 3 usando las instrucciones de la tabla 2.

Operaciones	Significado didáctico	Convenio de codificación

MULTI 64x, 68x
MOVER C8x, 6Cx
MULTI C8x, 70x
MULTI C8x, 74x
SUMAR 64x, C8x
	x_1 *= x_2;
	t1 = x_3;
	t1 *= x_4;
	t1 *= x_5;
	x_1 += t1;
		Se observa en las lineas 1, 3, 4 y 5 el uso de abreviaciones para simplificar que:
		x_1 *= x_2; corresponde $x_1 = x_1 + x_2$
		t_1 * = x_4; corresponde $t_1 = x_3 * x_4$
		t_1 * = e; corresponde $t_1 = t_1 * x_5$
		x_1 + = t_1; corresponde $x_1 = x_1 + t_1$
		La instrucción Mover se ha usado para asignar el valor inicial de la variable temporal t_1, de esta forma al escribir el fragmento de programa en esta tabla se ha utilizado las propiedad conmutativa de la multiplicación y suma.

Máquinas de una dirección

Este tipo de maquinas que funcionan bajo este esquema se les conoce como máquinas acumuladoras, en función de que el acumulador tiene la doble función de servir como eperando y destino y como uno de los operandos fuente. De esta forma se requiere especificar un operando explicito en cada instrucción y esta es la dirección del otro operando fuente.

Tabla 3. Instrucciones para una máquina de una dirección

Instrucción	Detalle
SUMAR dir	Sumar el valor almacenado en la posición de memoria dir al valor en el acumulador y almacenar el resultado en el acumulador. acum = acum + M[dir]
MULTI dir	Multiplicar el valor almacenado en la posición de memoria dir al valor en el acumulador y almacenar el resultado en el acumulador. acum = acum * M[dir]

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen X Edición No X – Año XXXX

CARGAR dir	Cargar el acumulador con el valor almacenado en la posición de memoria dir. acum = M[dir]
GUARDAR dir	Almacenar el valor en el acumulador en la posición de memoria dir. M[dir] = acum
Convenio de codificación: Puede resultar excepcionante la regla donde el acumulador es el operando destino.	

Caso 4. Usando las instrucciones de la tabla 3 se evalua la expresión siguiente:

$$x_1 = x_1 * x_2 + x_3 * x_4 * x_5;$$

Supongamos que las variables x_1, x_2, x_3, x_4 y x_5 están asociadas a las direcciones de memoria 64x, 68x, 6Cx, 70x, y 74x, respectivamente.

Solución aproximada para esta situación plateada usando las intrucciones de la tabla anterior

Operaciones	Significado didáctico	Convenio de codificación
CARGAR 64x	acum = x_1;	Se observa que no existe ninguna variable temporal explicita (t_1), para este caso el acumulador actua como área temporal necesaria, de tal forma que este puede actuar como temporal unico y expresiones mas dificiles pueden requerir de temporales explicitos.
MULTI 68x	acum *= x_2;	
GUARDAR 64x	x_1 = acum;	
CARGAR 6Cx	acum = x_3;	
MULTI 70x	acum *= x_4;	
MULTI 74x	acum *= x_5;	
SUMAR 64x	acum += x_1;	
GUARDAR 64x	x_1 = acum;	

Máquina de cero direcciones

Este escenario se leee algo contradictorio y se trata de maquinas que permiten a las instrucciones no tener direcciones explicitas, lo unico que debe conocerse es saber donde estan los dos operandos y donde almacenar el resultado, a este tipo se les llama tambien máquinas de pila, esto en razón de que los valores se almacenan en una pila interna; las operaciones toman valores de la pila y meten en ella los resultados. Se debe recordar que el concpeto de pila esta asociados a estructuras de datos abstractos (TDA), en los cuales solo dos operaciones se aplican a este TDA, ellas son el Meter y Sacar.

Tabla 4. Instrucciones para una máquina de cero dirección

Instrucción	**Detalle**

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen X Edición No X – Año XXXX

SUMAR	Extraer los dos valores de la cima de la pila, sumarlos y agregar el resultado en la pila Meter (sacar + sacar)
MULTI	Sacar los dos valores de la cima de la pila, umtiplicarlos y meter el resultado en la pila Meter (sacar * sacar)
METER dir	Meter el valor almacenado en la posición de memoria dir de la cima de la pila Meter (M[dir])
SACAR dir	Sacar el valor de la cima de la pila y almacenarlo en la posición de memoria dir. M[(dir]) = Sacar

Caso 5. Usando las instrucciones de la tabla 4 se evalua la expresión siguiente:

$$x_1 = x_1 * x_2 + x_3 * x_4 * x_5;$$

Suponemos que las variables x_1, x_2, x_3, x_4 y x_5 están asociadas a las direcciones de memoria 64x, 68x, 6Cx, 70x, y 74x, respectivamente.

Solución aproximada para esta situación plateada en la tabla anterior

Operaciones	Significado didáctico	Convenio de codificación
METER 64x METER 68x MULTI METER 6Cx METER 70x MULTI METER 74x MULTI SUMAR SACAR 64x	(x_1) (x_1) (x_2) $(x_1 * x_2)$ $(x_1 * x_2)$ (x_3) $(x_1 * x_2)$ (x_3) (x_4) $(x_1 * x_2)$ $(x_3 * x_4)$ $(x_1 * x_2)$ $(x_3 * x_4)$ (x_5) $(x_1 * x_2)$ $(x_3 * x_4 * x_5)$ $(x_1 * x_2 + x_3 * x_4 * x_5)$	Puede observarse que el planteamiento anterior no requiere de una variable temporal explicita, por el contrario es la pila la estructura de datos abstracta la que puede contener cualquier numero de variables temporales, y el numero de estas variables estará sujeto al tamaño de la pila.

Conclusiones

En la tecnologia de computadores el estudio de las máquinas requiere un grado de asimilación de porcesos complejos, esta complejidad se basa no en lo dificil, sino por lo abstracto, en tal sentido cada máquina tiene sus propias instrucciones, quienes estudiamos la estructura de Von Neumann como espacio de enseñanza de la arquitectura computacional vemos con preocupación el distanciamiento hacia el

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen X Edición No X – Año XXXX

hardware como recurso instruccional para un aprendizaje significativo en el campo de la Ingeniería.

En la estructura de Von Neumann se establecen tres unidades operacionales a saber; la Memoria, el Procesador y la E/S. En el Procesador subyace la Unidad de Control, elemento que contiene las instrucciones de la máquina tambien se le conocen como repertorio de instrucciones, la forma de trabajo de cada instrucción depende de su arquitectura, en el presente articulo se trajeron a colación cinco casos para denotar el funcionamiento de operaciones básicas de cálculo en este diseño computacional.

Cada instrucción tiene asociada una serie de operaciones binarias estandar y el almacenamiento de los resultados, a estas operaciones Maccabe (1995), Wakerly (1981), Hamacher, Vranesic y Zaky (1987) las denominan operaciones ternarias por la necesidad de especificar tres operandos, los dos de la operación binaria y un tercero que especifica donde se almacenará el resultado.

Hemos conceptualizado sobre las máquinas, asi en la máquina de tres direcciones los tres operandos aparecen en forma explicita en la instrucción, en una máquina de dos direcciones y una dirección existen un operando fuente y uno destino. En una máquina de una dirección el acumulador es el segundo operando fuente, tambien el destino, y en una máquina de cero direcciones la pila proporciona los dos operando fuentes y el destino.

Tal es el caso de las operaciones binarias (copiar, complemento, entro otras) en el contexto de la tecnología de computadores las instrucciones son los codigos elementales que proporcionan la riguidez funcional de un computador, usando los casos 1, 2 ,3 ,4 y 5 he tratado de esquematizar dos tendencias de funcionamiento, en la primera tendencia el numero de instruccciones necesarios parece aumentar cuando disminuye el numero de operandos explicitos en cada instrucción, para la segunda tendencia el tamaño de cada instrucción no es necesarrio especificar tantas direcciones.

El abordaje sobre los calculos sencillos en una computadora con arquietctura Von Neumenn se considera un tema de gran interes para el aruqitecto de computadoras, vista este equipo de computo como una unidad funcional que difieren en cuanto a su

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen X Edición No X – Año XXXX

diseño se trata entre si en cunato a tamaño, velocidad y costo, pero todas enfocadas al uso de instrucciones básicas o conunto de instrucciones reducidas o complejas.

Bibliografia

Hamacher, C. (1987). Organización de Computadoras, 2da Edición. McGraw-Hill, Mexico.

Maccabe, A. (1995). Sistemas Computacionales: Arquitectura y programamción de sistemas. Editorial McGraw-Hill/Irwin. Londres.

Kane, G. y Heinrich, J. (1992). MIPS RISC arquitectures. Prentice-Hall. USA.

Wakerly, J. (1981). Microcomputer Architecture and Programming. United Estates of America.