Apuntes de arquitectura de computadoras

Unidad 1 Modelo de arquitecturas de cómputo.

1.1 Modelos de arquitecturas de cómputo.

1.1.1 Arquitecturas Clásicas.

Estas arquitecturas se desarrollaron en las primeras computadoras electromecánicas y de tubos de vacío. Aun son usadas en procesadores empotrados de gama baja y son la base de la mayoría de las arquitecturas modernas

Arquitectura Mauchly-Eckert (Von Newman)

Esta arquitectura fue utilizada en la computadora ENIAC. Consiste en una unidad central de proceso que se comunica a través de un solo bus con un banco de memoria en donde se almacenan tanto los códigos de instrucción del programa, como los datos que serán procesados por este.

Esta arquitectura es la más empleada en la actualidad ya, que es muy versátil. Ejemplo de esta versatilidad es el funcionamiento de los compiladores, los cuales son programas que toman como entrada un archivo de texto conteniendo código fuente y generan como datos de salida, el código maquina que corresponde a dicho código fuente (Son programas que crean o modifican otros programas). Estos datos de salida pueden ejecutarse como un programa posteriormente ya que se usa la misma memoria para datos y para el código del programa.

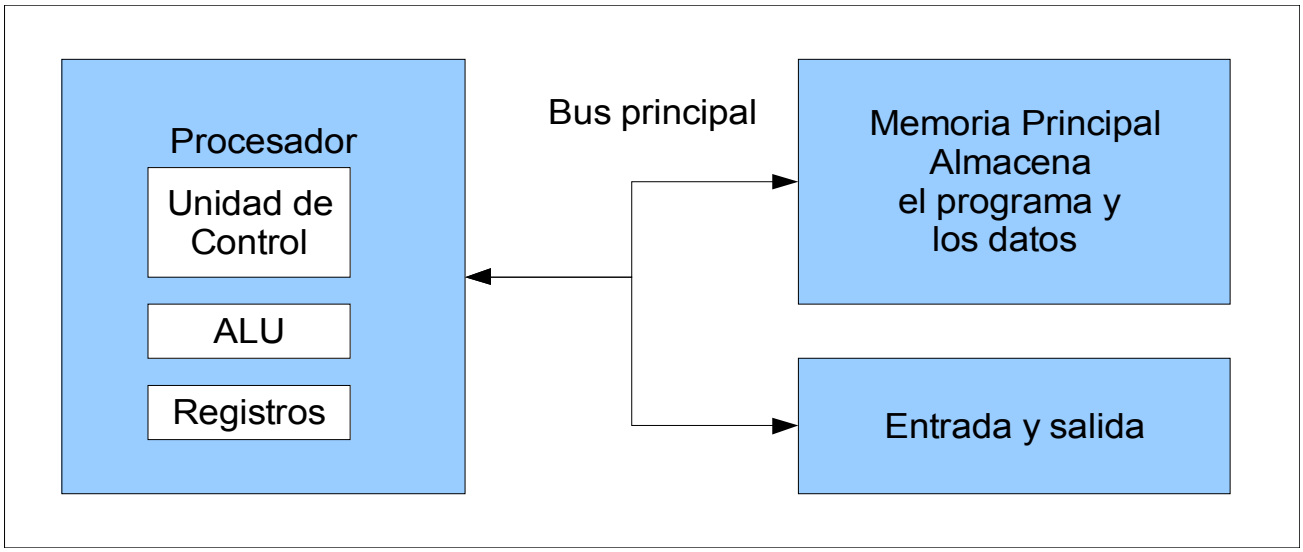

Figura 1.1.1.2 Diagrama a bloques de la arquitectura Von Newman.

La principal desventaja de esta arquitectura, es que el bus de datos y direcciones único se convierte en un cuello de botella por el cual debe pasar toda la información que se lee de o se escribe a la memoria, obligando a que todos los accesos a esta sean secuenciales. Esto limita el grado de paralelismo (acciones que se pueden realizar al mismo tiempo) y por lo tanto, el desempeño de la computadora. Este efecto se conoce como el cuello de botella de Von Newman

En esta arquitectura apareció por primera vez el concepto de programa almacenado. Anteriormente la secuencia de las operaciones era dictada por el alambrado de la unidad de control, y cambiarla

implicaba un proceso de recableado laborioso, lento(hasta tres semanas) y propenso a errores. En esta arquitectura se asigna un código numérico a cada instrucción. Dichos códigos se almacenan en la misma unidad de memoria que los datos que van a procesarse, para ser ejecutados en el orden en que se encuentran almacenados en memoria. Esto permite cambiar rápidamente la aplicación de la computadora y dio origen a las computadoras de propósito general

Mas a detalle, el procesador se subdivide en una unidad de control (C.U.), una unidad lógica aritmética (A.L.U.) y una serie de registros. Los registros sirven para almacenar internamente datos y estado del procesador. La unidad aritmética lógica proporciona la capacidad de realizar operaciones aritméticas y lógicas. La unidad de control genera las señales de control para leer el código de las instrucciones, decodificarlas y hacer que la ALU las ejecute.

Arquitectura Harvard

Esta arquitectura surgió en la universidad del mismo nombre, poco después de que la arquitectura Von Newman apareciera en la universidad de Princeton. Al igual que en la arquitectura Von Newman, el programa se almacena como un código numérico en la memoria, pero no en el mismo espacio de memoria ni en el mismo formato que los datos. Por ejemplo, se pueden almacenar las instrucciones en doce bits en la memoria de programa, mientras los datos de almacenan en ocho bits en una memoria aparte.

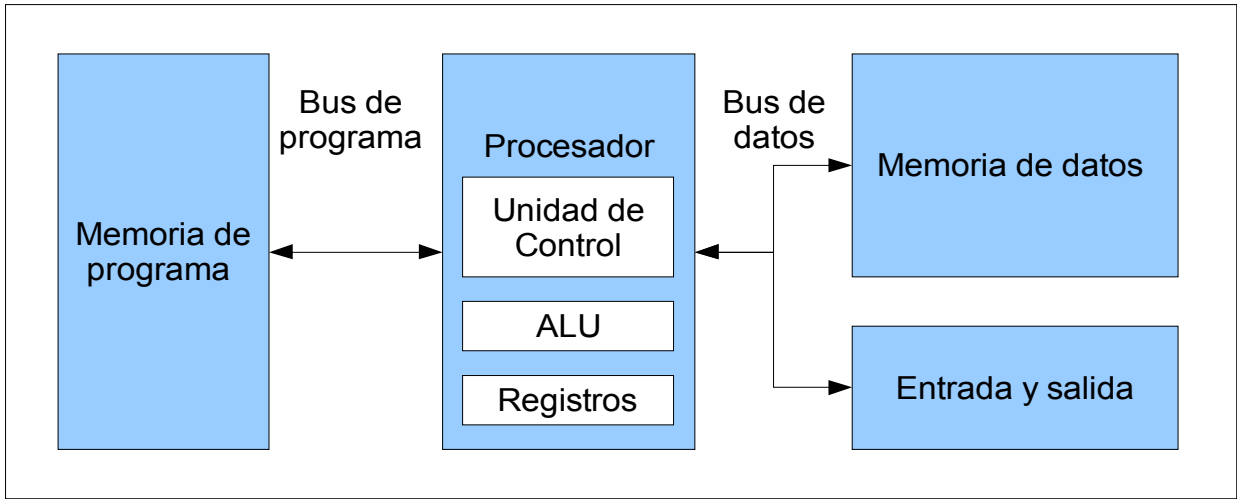

Figura 1.1.1.2 Diagrama a bloques de la arquitectura Harvard

El hecho de tener un bus separado para el programa y otro para los datos permite que se lea el código de operación de una instrucción, al mismo tiempo se lee de la memoria de datos los operados de la instrucción previa. Así se evita el problema del cuello de botella de Von Newman y se obtiene un mejor desempeño.

En la actualidad la mayoría de los procesadores modernos se conectan al exterior de manera similar a a la arquitectura Von Newman, con un banco de memoria masivo único, pero internamente incluyen varios niveles de memoria cache con bancos separados en cache de programa y cache de datos, buscando un mejor desempeño sin perder la versatilidad.

1.1.2 Arquitecturas Segmentadas.

Las arquitecturas segmentadas o con segmentación del cauce buscan mejorar el desempeño realizando paralelamente varias etapas del ciclo de instrucción al mismo tiempo. El procesador se divide en varias unidades funcionales independientes y se dividen entre ellas el procesamiento de las instrucciones. Para

comprender mejor esto, supongamos que un procesador simple tiene un ciclo de instrucción sencillo consistente solamente en una etapa de búsqueda del código de instrucción y en otra etapa de ejecución de la instrucción. En un procesador sin segmentación del cauce, las dos etapas se realizarían de manera secuencial para cada una de la instrucciones, como lo muestra la siguiente figura.

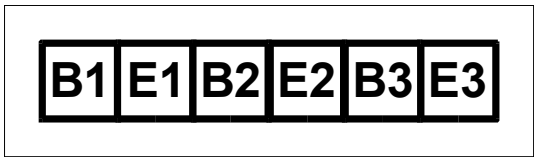

Figura 1.1.2.1 Búsqueda y ejecución en secuencia de tres instrucciones en un procesador sin segmentación del cause

En un procesador con segmentación del cause, cada una de estas etapas se asigna a una unidad funcional diferente, la búsqueda a la unidad de búsqueda y la ejecución a la unidad de ejecución. Estas unidades pueden trabajar en forma paralela en instrucciones diferentes. Estas unidades se comunican por medio de una cola de instrucciones en la que la unidad de búsqueda coloca los códigos de instrucción que leyó para que la unidad de ejecución los tome de la cola y los ejecute. Esta cola se parece a un tubo donde las instrucciones entran por un extremo y salen por el otro. De esta analogía proviene el nombre en ingles: Pipelining o entubamiento.

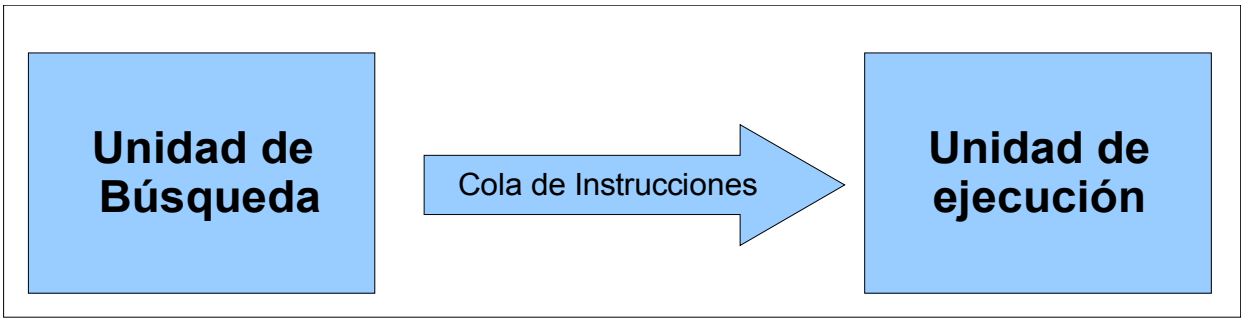

Figura 1.1.2.3 Comunicación entre las unidades en un procesador con segmentación de cauce.

Completando el ejemplo anterior, en un procesador con segmentación, la unidad de búsqueda comenzaría buscando el código de la primera instrucción en el primer ciclo de reloj. Durante el segundo ciclo de reloj, la unidad de búsqueda obtendría el código de la instrucción 2, mientras que la unidad de ejecución ejecuta la instrucción 1 y así sucesivamente. La siguiente figura muestra este proceso.

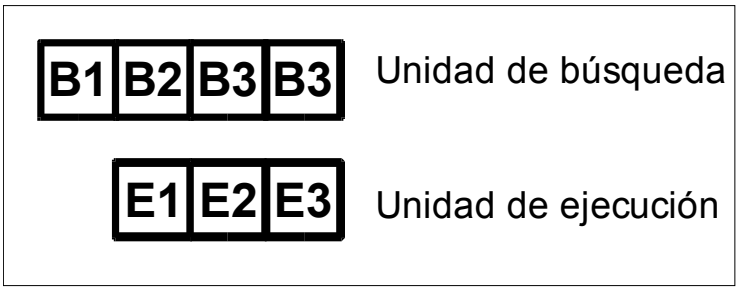

Figura 1.1.2.3 Búsqueda y ejecución en secuencia de tres instrucciones en un procesador con segmentación del cause

En este esquema sigue tomando el mismo numero de ciclos de reloj (el mismo tiempo), pero como se

trabaja en varias instrucciones al mismo tiempo, el número promedio de instrucciones por segundo se multiplica. La mejora en el rendimiento no es proporcional al numero de segmentos en el cauce debido a que cada etapa no toma el mismo tiempo en realizarse, además de que se puede presentar competencia por el uso de algunos recursos como la memoria principal. Otra razón por la que las ventajas de este esquema se pierden es cuando se encuentra un salto en el programa y todas las instrucciones que ya se buscaron y se encuentran en la cola, deben descartarse y comenzar a buscar las instrucciones desde cero a partir de la dirección a la que se salto. Esto reduce el desempeño del procesador y aún se investigan maneras de predecir los saltos para evitar este problema.

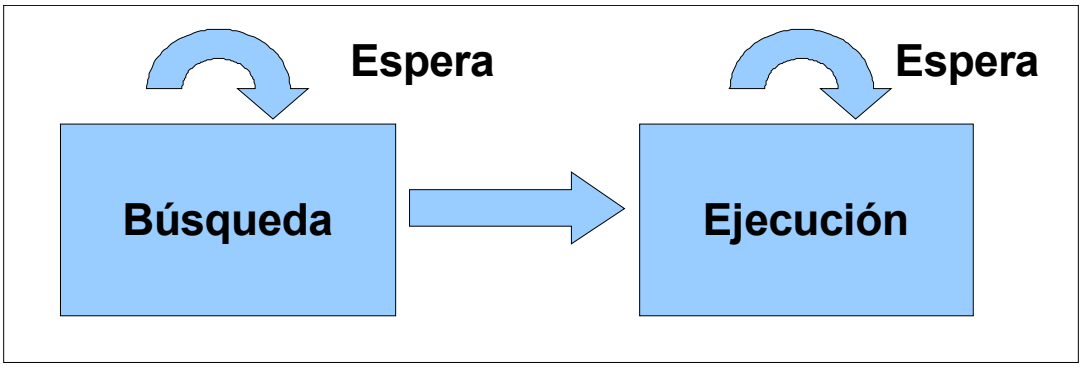

Figura 1.1.2.4 Consecuencias de la competencia por un recurso

1.1.3 Arquitecturas de multiprocesamiento.

Cuando se desea incrementar el desempeño más aya de lo que permite la técnica de segmentación del cauce (limite teórico de una instrucción por ciclo de reloj), se requiere utilizar más de un procesador para la ejecución del programa de aplicación.

Las CPU de multiprocesamiento se clasifican de la siguiente manera (Clasificación de Flynn):

- SISO – (Single Instruction, Single Operand) computadoras Monoprocesador
- SIMO – (Single Instruction, Multiple Operand) procesadores vectoriales, Exenciones MMX
- MISO – (Multiple Instruction, Single Operand) No implementado
- MIMO – (Multiple Instruction, Multiple Operand) sistemas SMP, Clusters, GPUs

Procesadores vectoriales – Son computadoras pensadas para aplicar un mismo algoritmo numérico a una serie de datos matriciales, en especial en la simulación de sistemas físicos complejos, tales como simuladores para predecir el clima, explosiones atómicas, reacciones químicas complejas, etc., donde los datos son representados como grandes números de datos en forma matricial sobre los que se deben se aplicar el mismo algoritmo numérico.

La mayoría de los procesadores modernos incluye algunas instrucciones de tipo vectorial, tales como las extensiones al conjunto de instrucciones tales como MMX y SSE. Estas instrucciones les permiten procesar flujos multimedia más eficientemente.

Los Procesadores Digitales de Señales (DSP), son procesadores especializados en el procesamiento de señales tales como audio, vídeo, radar, sonar, radio, etc. Cuentan con instrucciones tipo vectorial que los hace muy aptos para dicha aplicación. Suelen utilizarse en conjunto con un microcontrolador en dispositivos como reproductores de audio, reproductores de dvd y Blueray, teléfonos celulares, sistemas de entretenimiento, sistemas de adquisición de datos, instrumentos médicos, controles industriales, etc.

En los sistemas SMP (Simetric Multiprocesesors), varios procesadores comparten la misma memoria principal y periféricos de I/O, Normalmente conectados por un bus común. Se conocen como simétricos, ya que ningún procesador toma el papel de maestro y los demás de esclavos, sino que todos tienen derechos similares en cuanto al acceso a la memoria y periféricos y ambos son administrados por el sistema operativo.

Pueden formarse con varios núcleos en un solo circuito integrado o con varios circuitos integrados en una misma tarjeta madre. La primera opción ha sido popularizada al hacerse más económicos los procesadores multinucleo de los principales fabricantes y con su uso en sistemas de gama media y baja, e inclusive en teléfonos celulares y tabletas. La segunda opción fue la que se uso en un principio y sigue siendo usada en en estaciones de trabajo y en servidores de alto rendimiento debido a que incrementa el poder computacional del sistema, pero también incrementa considerablemente el costo del sistema.

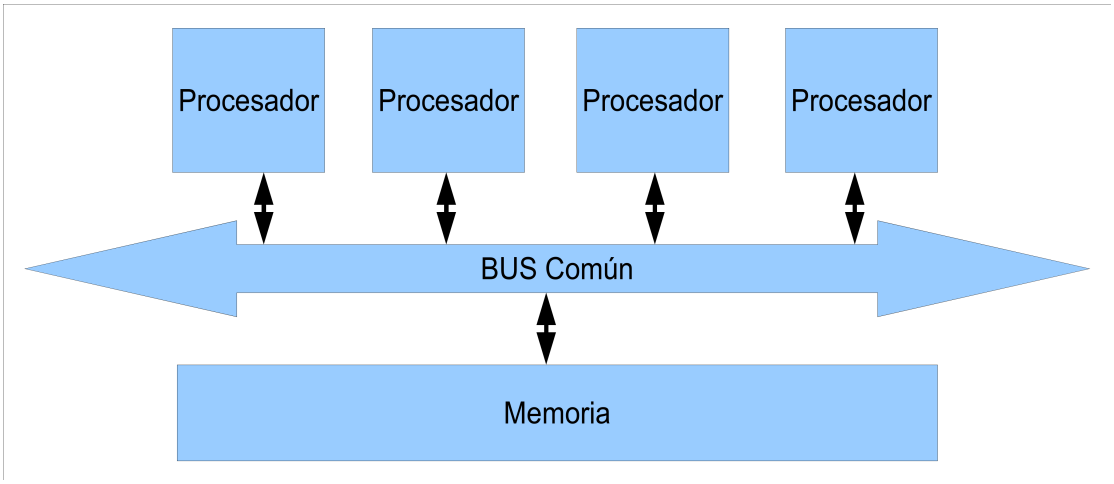

Figura 1.1.3.1 Diagrama a bloques de un sistema multiprocesador simétrico

Los Clusters son conjuntos de computadoras independientes conectadas en una red de área local o por un bis de interconexión y que trabajan cooperativamente para resolver un problema. Es clave en su funcionamiento contar con un sistema operativo y programas de aplicación capaces de distribuir el trabajo entre las computadoras de la red. Este tipo de computadora paralela se ha vuelto muy popular por que permite usar los avances en los procesadores comerciales que tienen una muy buena relación costo rendimiento y se puede incorporar rápidamente los avances que proporciona las nuevas tecnologías en cuanto es económicamente viable.

Sin embargo, se debe tener cuidado al implementar la aplicación, ya que si los datos que hay que pasar de un procesador a otro son demasiados, el tiempo empleado en pasar información de un nodo a otro puede sobrepasar a la ganancia que se tiene al dividir el trabajo entre varios procesadores.

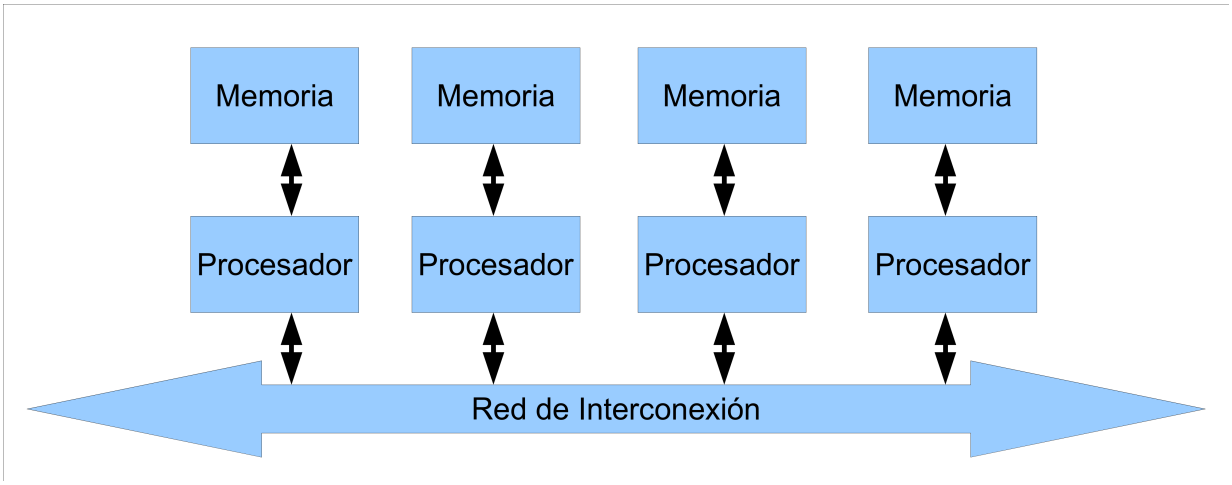

Figura 1.1.3.2 Diagrama a bloques de un cluster

Las unidades de procesamiento gráfico (Graphics Processing Unit GPU) – sistemas diseñados originalmente para el procesamiento de Gráficos, con múltiples procesadores vectoriales sencillos compartiendo la misma memoria, la cual también puede ser accedida por el CPU. Por la gran cantidad de núcleos con los que cuenta, logran un excelente desempeño al ejecutar algoritmos que se adaptan a ser paralelizados, a tal grado que muchas de las supercomputadoras más rápidas de la actualidad utilizan estos procesadores, y los fabricantes de tarjetas gráficas producen versiones de sus productos especializadas en acelerar los cálculos de propósito general.

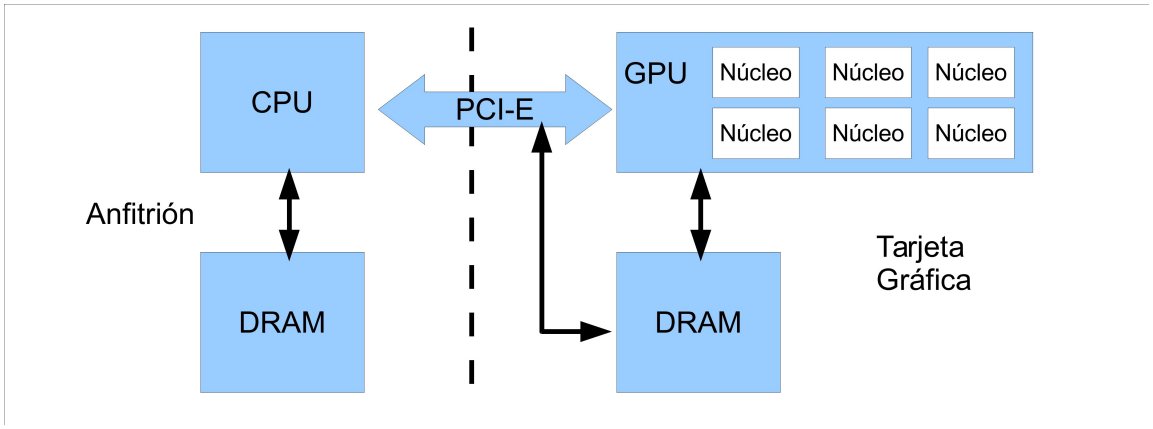

Figura 1.1.3.3 Diagrama a bloques de una unidad de procesamiento gráfico

1.2 Análisis de los componentes.

1.2.1 CPU.

1.2.1.1 Arquitecturas.

Ademas de las arquitecturas clásicas mencionadas anteriormente, en la actualidad han aparecido arquitecturas híbridas entre la Von Newman y la Harvard, buscando conservar la flexibilidad, pero mejorando el rendimiento.

Uno de los cambios más importante de los últimos años en diseño de las computadoras se dio durante

los años 1980s, con la aparición de la corriente de diseño conocida como computadoras de conjunto reducido de instrucciones (RISC, por sus siglas en ingles). Esta escuela pretende aplicar un enfoque totalmente distinto al tradicional hasta entonces, que paso a conocerse como computadoras de conjunto complejo de instrucciones (CISC) para diferenciarla de la nueva tendencia.

La tendencia tradicional, representada por las arquitecturas CISC (Complex Instruction Set Computers) se caracterizan por tener un número amplio de instrucciones y modos de direccionamiento. Se implementan instrucciones especiales que realizan funciones complejas, de manera que un programador puede encontrar con seguridad, una instrucción especial que realiza en hardware la función que el necesita. El número de registros del CPU es limitado, ya que las compuertas lógicas del circuito integrado se emplean para implementar las secuencias de control de estas instrucciones especiales.

Al investigar las tendencias en la escritura de software científico y comercial al inicio de los 80, ya se pudo observar que en general ya no se programaba mucho en ensamblador, sino en lenguajes de alto nivel, tales como C. Los compiladores de lenguajes de alto nivel no hacían uso de las instrucciones especiales implementadas en los procesadores CISC, por lo que resultaba un desperdicio de recursos emplear las compuertas del circuito de esta forma. Por lo anterior, se decidió que era mejor emplear estos recursos en hacer que las pocas instrucciones que realmente empleaban los compiladores se ejecutaran lo más rápidamente posible. Así surgió la escuela de diseño RISC (Reduced Instruction Set Computers) donde solo se cuenta con unas pocas instrucciones y modos de direccionamiento, pero se busca implementarlos de forma muy eficiente y que todas las instrucciones trabajen con todos los modos de direccionamiento. Ademas, se observo que una de las tareas que tomaban más tiempo en ejecutarse en lenguajes de alto nivel, era el pasar los parámetros a las subrutinas a través de la pila. Como la forma más rápida de hacer este paso es por medio de registros del CPU, se busco dotarlo con un amplio número de registros, a través de los cuales se pueden pasar dichos parámetros.

1.2.1.2 Tipos.

Los CPUs modernos pueden clasificarse de acuerdo a varias características, tales como: el tamaño del ALU o del Bus de conexión al exterior (8, 16, 32, 64 bits),si tienen cauce segmentado o no segmentado, si con tipo CISC o RISC, Von Newan o Harvard y si solo tienen instrucciones enteras o implementan también instrucciones de punto flotante

1.2.1.3 Características.

Las características más importantes a considerar al escoger un CPU para usarlo en una aplicación, son:

•Modelo del programador (Conjunto de registros que el programador puede utilizar), forman el modelo mental del CPU que el programador utiliza al programar en ensamblador.

•Conjunto de instrucciones que puede ejecutar el CPU

•Modos de direccionamiento que pueden usarse para obtener los operandos de las instrucciones.

•Ciclo de instrucción (el conjunto de pasos que realiza el CPU para procesar cada instrucción)

•Buses de interconexión, usados para que el CPU lea y escriba a la memoria y a los dispositivos de entrada y salida.

1.2.1.4 Funcionamiento.

Debido a la gran variedad de CPU disponibles comercialmente, se explicara el funcionamiento de un unidad central de proceso imaginaria muy simple, pero que resume el funcionamiento básico de la

mayoría de los CPUs. Este CPU es similar a las primeras computadoras existentes en los años 1950s (Basado en el diseño explicado en Mano, M. Morris. Arquitectura de computadoras. Pearson 1994).

Esta computadora contara con una memoria de 4096 palabras de 16 bits cada una. Esto corresponde a un bus de direcciones de 12 bits y un bus de datos de 16 bits). En cada localidad de memoria se podrá almacenar un entero de 16 bits o el código de una instrucción, también de 16 bits.

Todos los CPU tienen como función principal la ejecución de un programa acorde a la aplicación del mismo. Un programa es un conjunto de instrucciones almacenadas de acuerdo al orden en que deben ejecutarse. Por lo tanto, toda computadora debe ser capaz de procesar las instrucciones de su programa en un ciclo de instrucción, consistente en un número de etapas que varia con cada CPU, pero que tradicionalmente han sido tres:

1-**Búsqueda** del código de Instrucción. Esta consiste en leer de la memoria cual será la siguiente instrucción a ejecutar, la cual esta almacenada en forma de un código numérico que indica cual de todas las operaciones que puede realizar el CPU sera la siguiente y con que operandos se ejecutara.

2- **Decodificación**. Consiste en tomar el código numérico e identificar a cual de las operaciones que puede realizar el CPU corresponde dicho código. El proceso contrario, la codificación, consiste en conociendo la instrucción, determinar el número que la va a representar. Esta etapa usualmente se realiza con un decodificador binario.

3- **Ejecución**. En esta etapa se lleva a cabo la operación sobre los datos que se vallan a procesar. En general, la unidad de control (CU) genera las señales de control necesarias para llevar los datos a las entradas de la Unidad Aritmética Lógica, la cual efectuará las operaciones aritméticas y lógicas. Posteriormente, la unidad de control generara las señales de control necesarias para transferir la salida de la Unidad Aritmética Lógica al registro donde serán almacenados los resultados para su uso posterior.

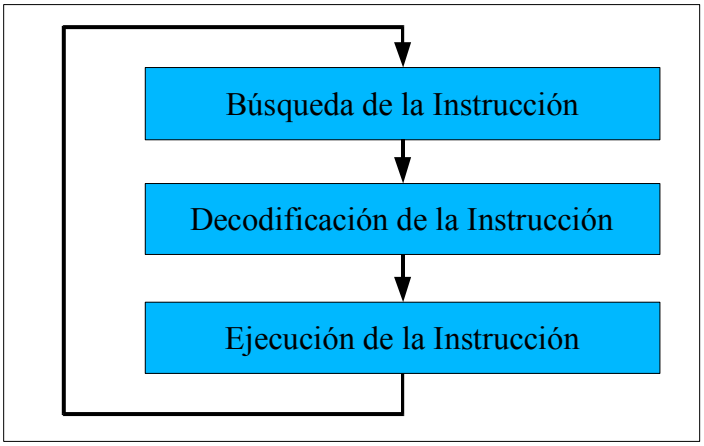

Figura 1.2.1.4.1 Ciclo de instrucción

Es importante recordar que cada instrucción del programa se almacena en memoria como un número binario. Este número se conoce como código de instrucción, y usualmente se divide en al menos dos campos: un código de operación (Opcode) y un número que representa al operando u operandos de la instrucción. En el caso de la computadora imaginaria que estamos estudiando, se almacena cada instrucción en una de las 4096 palabras de memoria de 16 bits cada una. Se utiliza un formato de un solo operando, con un segundo operando en el acumulador cuando es necesario. Los cuatro bits más significativos de los dieciséis bits de la palabra se dedican a almacenar el código de operación. Los doce bits menos significativos se dedican a almacenar la dirección del operando.

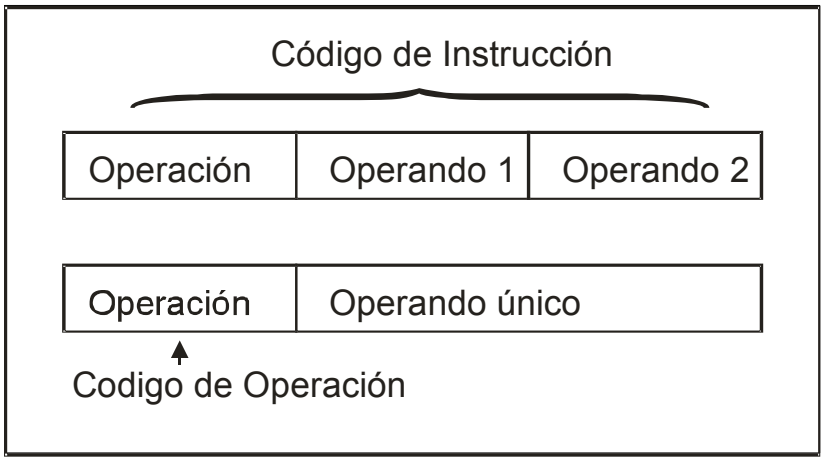

Figura 1.2.1.4.2 Codificación de las instrucciones

La siguiente tabla resume los códigos de operación de la computadora de ejemplo.

Código de Operación	Instrucción	Operación
0h	LOAD (Carga)	ACC<-[M]
1h	STORE (Almacena)	[M]<-ACC
2h	ADD (Suma)	ACC<-ACC+[M]
3h	ADC (Suma con Acarreo)	ACC<-ACC+[M]+C
4h	SUB (Resta)	ACC<-ACC+-M]
5h	OR (Or Bit a Bit)	ACC<-ACC or [M]
6h	AND (And bit a Bit)	ACC<-ACC and [M]
7h	XOR (Xor Bit a Bit)	ACC<-ACC xor [M]
8h	SHL (Corrimiento a la Izquierda)	ACC<-ACC << 1
9h	SHR (Corrimiento a la derecha)	ACC<-ACC >> 1
Ah	BRA Bifurcación o salto	PC<-M
Bh	BRZ (Bifurca si es Cero)	Si Z==1 => PC<-M
Ch	BRC (Bifurca si hay Acarreo)	Si C==1 => PC<-M
Dh	BRO (Bifurca si hay Sobreflujo)	Si O==1 => PC<-M
Eh	LDI (Carga Constante Inmediata)	ACC <-[PC]; PC<-PC+1
Fh	STOP	Detener la ejecución

Tabla 1.2.1.4.1 Códigos de operación para la computadora de ejemplo

Las partes del CPU de la computadora imaginaria son:
ACC – Acumulador, se usará para almacenar uno de los operandos y el resultado de varias de las instrucciones
MAR – (Memory Address Register) Registro de dirección de memoria, selecciona a que localidad de memoria se va a leer o a escribir.
MBR – (Memory Bus Register) Registro de bus de memoria. A través de él se lee y se escriben los

datos.
PC – (Program Counter) El contador de programa almacena la dirección de la siguiente instrucción a buscar. Por esta razón también es conocido como apuntador de instrucciones.
IR - Registro de instrucción, guarda el código de la instrucción que se esta ejecutando.
Flags – Registro de Banderas, agrupa a todas las banderas de la ALU en un registro, en el caso de nuestra computadora imaginaria, las banderas disponibles serán: Z – Bandera de Cero, se pone en uno cuando todos los bits del resultado son cero; O – Sobreflujo, se pone en uno cuando el resultado de la ultima operación se sale de el rango de los números de 16 bits con signo; C – Acarreo, se enciende cuando el resultado de la ultima operación se sale del rengo de los números de 16 bits sin signo.

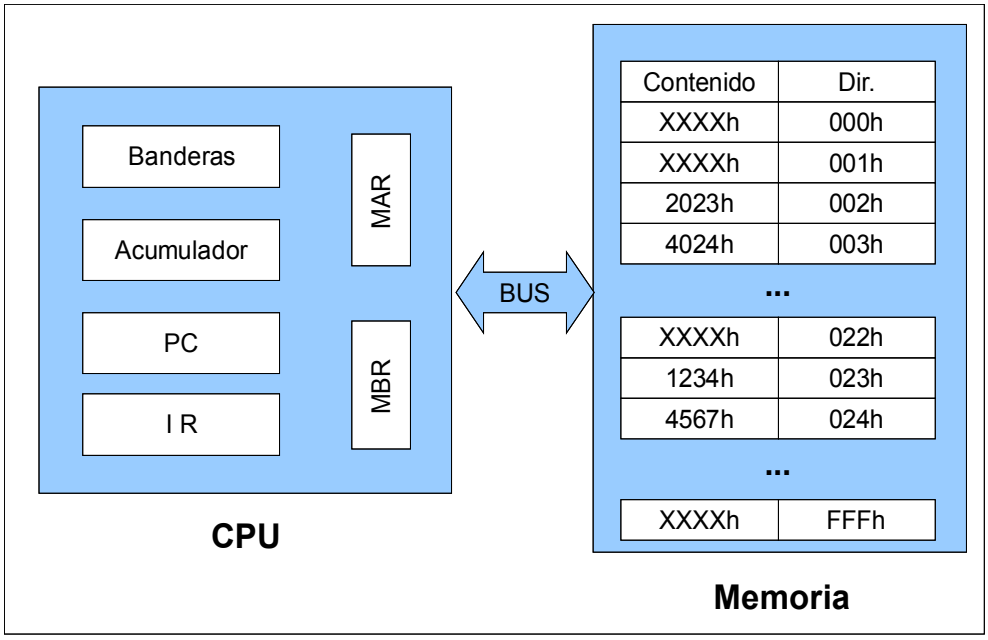

Figura 1.2.1.4.2 Diagrama a bloques del CPU a estudiar

Como se explico anteriormente, el funcionamiento del CPU se basa en los pasos del ciclo de instrucción, consistentes en búsqueda, decodificación y ejecución de la instrucción. Comenzaremos revisando los pasos correspondientes a la búsqueda de la instrucción.

El registro PC contiene la dirección de la localidad de memoria que contiene el código de instrucción de la siguiente instrucción a ejecutar. Como la etapa de búsqueda consistirá básicamente en leer este código y almacenarlo en el registro IR para su posterior uso en las etapas de decodificación y ejecución, el contenido de PC se copia al MAR para poder leer esa localidad de memoria. Se lee la memoria y el resultado de dicha lectura se copia del MBR al IR. Finalmente, se incrementa el PC para que en el siguiente ciclo de instrucción se lea la instrucción de la localidad de memoria consecutiva.
Resumiendo estas operaciones en lenguaje de transferencia de registros:
MAR<-PC
IR<-[MAR]
PC<-PC+1
En la etapa de decodificación simplemente se separan los códigos de operación de los operandos. Por ejemplo, la instrucción LOAD 023h se codificaría como 0023h, siendo 0h el código de operación y 023h el operando. Además, la unidad de control deberá identificar que al opcode 0 corresponde a la instrucción LOAD para que en la siguiente etapa se realicen las operaciones correspondientes a esta instrucción.
En cuanto a la etapa de ejecución, los pasos realizados en esta etapa varían dependiendo del código de operación leído en la etapa de búsqueda. Por ejemplo, si el código leído es un 0, que corresponde con

una instrucción LOAD, la etapa de ejecución consistirá en copiar la parte de la dirección del operando en el registro MAR para poder leer la localidad en donde se encuentra el operando. Se lee el operando de memoria y el dato leído se copia del MBR al acumulador. Resumiendo dichas operaciones en lenguaje de transferencia de registros se tiene:
MAR<-IR(M) //IR(M) representa los bits del registro IR que almacenan la dirección del operando
ACC<-MBR

Para el código de operación 2, correspondiente a la instrucción ADD, también se transfiere la dirección del operando al MAR y se lee el contenido de la memoria, pero en vez de enviarse directamente del MBR al acumulador, se envía al ALU para que se sume con el contenido del acumulador y posteriormente se almacene el resultado de la suma en el acumulador. Resumiendo en lenguaje de transferencia de registros:
MAR<-IR(M) //IR(M) representa los bits del registro IR que almacenan la dirección del operando
ACC<-ACC+MBR
La columna operación de la Tabla 1.2.1.4.1 proporciona un resumen el lenguaje de transferencia de registros de las operaciones correspondientes a cada código de operación.