CAPÍTULO 5

PROCESADOR PARTE I

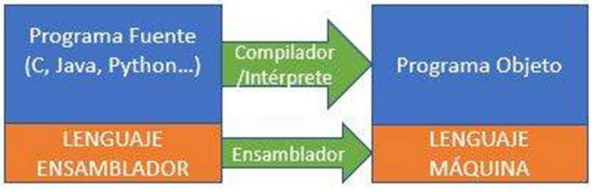
1. Motivación y necesidad del lenguaje máquina

Los lenguajes de programación son herramientas que permiten a los programadores escribir y desarrollar aplicaciones y sistemas informáticos. Existen diferentes categorías de lenguajes de programación, y algunos de ellos se utilizan específicamente para el desarrollo de aplicaciones en ensamblador. Algunos de los lenguajes de programación relevantes para un curso de ensamblador incluyen:

1. Lenguaje de Máquina: Es el lenguaje de programación más básico y cercano a la arquitectura de la CPU.
2. Ensamblador: Es un lenguaje de programación de bajo nivel que permite escribir código fácilmente traducible a lenguaje de máquina.
3. Lenguajes de Alto Nivel: Son lenguajes de programación más abstractos y cercanos al lenguaje humano, como C, Python, Java, etc.
4. Lenguajes de Programación Específicos: Son lenguajes de programación diseñados para un propósito específico, como SQL para la gestión de bases de datos, HTML para la creación de páginas web, etc.

2. El lenguaje máquina

- Programa fuente: Es un archivo de texto que contiene el código escrito en un lenguaje de programación, tal como C, Python, Java, etc. El programa fuente no puede ser ejecutado directamente por la computadora, sino que debe ser compilado o interpretado antes de ser ejecutado.
- Lenguaje ensamblador: Es un lenguaje de programación de bajo nivel que permite escribir código que sea fácilmente traducido a lenguaje de máquina. Un ensamblador traduce cada instrucción escrita en lenguaje ensamblador a un código binario que puede ser entendido y ejecutado por la CPU.
- Ensamblador: Aplicación encargada de traducir el lenguaje ensamblador al lenguaje máquina.
- Programa objeto: Es un archivo generado a partir del programa fuente que contiene el código traducido a un formato ejecutable por la CPU. Este archivo es generado por un compilador o un intérprete y contiene instrucciones en lenguaje de máquina o ensamblador. El programa objeto puede ser ejecutado directamente por la computadora.
- Lenguaje máquina: Instrucciones que ejecuta el procesador. Son instrucciones nativas del procesador, es decir, que son diferentes por cada familia/fabricante.



Lenguaje de Alto nivel

Es un lenguaje de programación que utiliza una sintaxis más cercana al lenguaje humano y se basa en conceptos abstractos, como variables, estructuras de control de flujo y funciones. Ejemplos incluyen Python, Java y C++.

El lenguaje ensamblador

Es un lenguaje de programación que se utiliza para programar directamente en lenguaje máquina. El ensamblador permite un control más fino de la CPU y el sistema, pero requiere un conocimiento profundo de la arquitectura de la CPU y de la memoria.

El lenguaje máquina

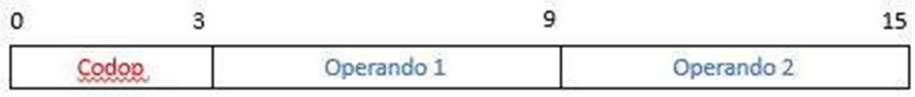
Es un lenguaje de programación que se utiliza para programar directamente en código binario, en forma de 0s y 1s. Es el lenguaje de más bajo nivel y no es legible por los humanos, por lo que se requiere de un ensamblador o un compilador para escribir programas en él.

3. Instrucciones Máquina

Las instrucciones de máquina son las órdenes que la CPU puede ejecutar directamente, estas instrucciones están escritas en un código binario que la CPU puede entender y procesar. Las instrucciones de máquina pueden realizar una amplia variedad de tareas, como:

- Cargar y almacenar valores en la memoria.
- Realizar cálculos y comparaciones.
- Saltar a diferentes secciones del código, etc.

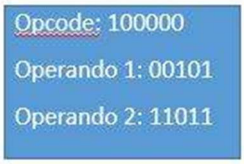
Cada instrucción de máquina está compuesta por una serie de bits que representan un "código op" o "codop", que identifica la operación a realizar, y por otros bits que representan los operandos necesarios para llevar a cabo la operación. Por ejemplo, para 16 bits tendríamos:



Cada instrucción se divide en los siguientes elementos (ejemplo con 16 bits visto arriba):

- El codop es el código binario que define la operación a realizar por la CPU.
- Los operandos pueden ser de origen y de destino. Pueden estar localizados en:
    - La memoria principal.
    - Los registros.
    - Un dispositivo de E/S.

Así por ejemplo una instrucción de máquina para sumar dos números podría tener la siguiente estructura:



El repertorio de instrucciones de una CPU es un conjunto de ordenes que la CPU es capaz de ejecutar para realizar una amplia variedad de tareas lo que incluye:

- Operaciones aritmético-lógicas

- Operaciones de transferencia de datos
- Operaciones de control de flujo
- Operaciones de memoria

El número y tipo de instrucciones que soporta una CPU puede variar dependiendo del tipo de CPU y de su arquitectura, siendo este repertorio esencial en la velocidad y eficiencia de la CPU. Cada instrucción tiene asociado su propio codop, así como su propio nemotécnico o nemónico. Por ejemplo:

Codop	Nemónico	Operación
0001	ADD	Sumar
0010	SUB	Restar
0011	LD	Cargar

Los operandos son los valores o direcciones de memoria utilizadas en las operaciones de instrucciones máquina, cada una puede tener uno o mas operandos dependiendo de su naturaleza los cuales pueden ser valores constantes almacenados en la propia instrucción o direcciones de memoria. Los más frecuentes:

- Direcciones de memoria: que reflejan una dirección donde almacenamos un dato de interés conectadas entre paréntesis y expresadas en hexadecimal.
- Números: como parte del cómputo, aunque la ALU trabaje en binario se pueden representar en base 6.
- Caracteres: Utilizados en secuencia para representar texto

Código ASCII

Su nombre es el acrónimo de American Standard Code for Information Interchange. Es decir, "Código Americano Estándar para el Intercambio de Información". Este acrónimo se pronuncia como "Aski". Define un conjunto de caracteres y símbolos que se pueden utilizar para representar texto, incluyendo letras mayúsculas y minúsculas, números, signos de puntuación y caracteres de control tales como el salto de línea y la tabulación.

Es un estándar ampliamente utilizado y muchos otros códigos de representación de caracteres, como Unicode, han evolucionado a partir de ASCII.

4. El micro Z80

- Un registro es una pequeña área de memoria interna en un procesador donde se almacenan valores temporales y se realizan operaciones.
- Los registros son mucho más rápidos que la memoria RAM externa, por lo que el procesador los utiliza para almacenar y manipular los datos más críticos.
- Los registros son un elemento clave en la arquitectura de un procesador.
- Cada procesador tiene un número diferente de registros dependiendo de su arquitectura, cada uno con diferentes usos y propósitos.
- Cada registro tiene una función específica y su contenido puede ser leído y escrito por el procesador en una operación de un ciclo de reloj.

El Z80 tiene 16 registros de 8 bits (1 byte) y 4 registros de 16 bits (2 bytes). A su vez, estos registros se dividen en dos grupos: 12 registros de propósito general y 8 dedicados (4 de 16 bytes).

Z80 Registros de propósito general

Los registros de propósito general sirven para almacenar datos sobre los que operar sin tener que referenciar a la memoria. Están divididos en dos juegos, normales (sin apóstrofe) y '(con apóstrofe). Son:

- B, C, D, E, H, L, B', C', D', E', H' y L'.
- El procesador sólo puede acceder a uno de los juegos al mismo tiempo.

Son todos de 8 bits, aunque se pueden emparejar para trabajar con datos de 16 bits de la siguiente forma:

- BC, DE, HL, B'C', D'E' y H'L'.

Existe una instrucción específica (EXX) para cambiar el juego de registros con el que se trabaja en un instante dado.

Z80 Registros dedicados

Los registros dedicados son: A, A', F, F', PC, IX, IY y SP. Vamos a comentar cada uno de ellos:

A. Registro A (Acumulador):
   Es el registro principal utilizado para operaciones aritméticas y lógicas. También se utiliza para el intercambio de datos con la memoria y los puertos de entrada/salida.
B. Registros de Propósito General (B, C, D, E, H, L):
   Estos registros se utilizan para almacenar datos temporales y direccionamiento. Pueden ser utilizados individualmente o en parejas (BC, DE, HL) para operaciones más complejas.
C. Registro F (Flag):
   Almacena los bits de estado después de realizar operaciones aritméticas y lógicas. Los bits de este registro indican resultados como carry, zero, signo, etc.
D. Registros de Propósito Alternativo (A', B', C', D', E', H', L'):
   Se utilizan para almacenar valores temporales, especialmente durante llamadas a subrutinas, interrupciones, o durante la ejecución de ciertas instrucciones.
E. Registro de Contador de Programa (PC):
   Contiene la dirección de la próxima instrucción que se va a ejecutar. Se incrementa automáticamente después de cada instrucción ejecutada.

F. Registro de Puntero de Pila (SP):

Apunta a la posición actual de la pila en la memoria. Se utiliza para gestionar la pila durante las operaciones de llamada a subrutinas e interrupciones.

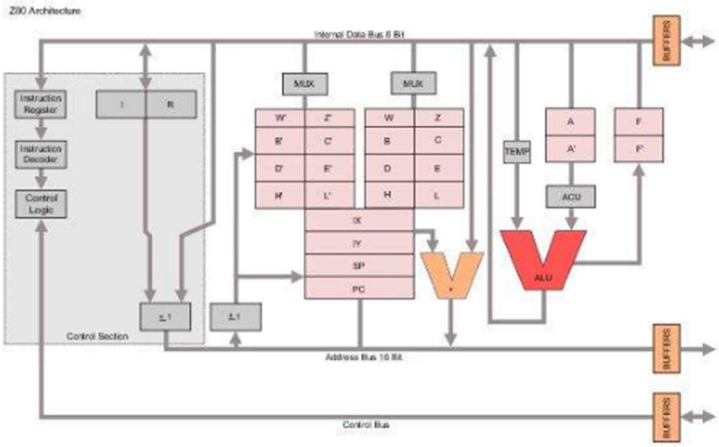
G. Registro de Índice (IX, IY):

Se utilizan para direccionamiento indirecto y para acceder eficientemente a datos en estructuras de datos como matrices o tablas.

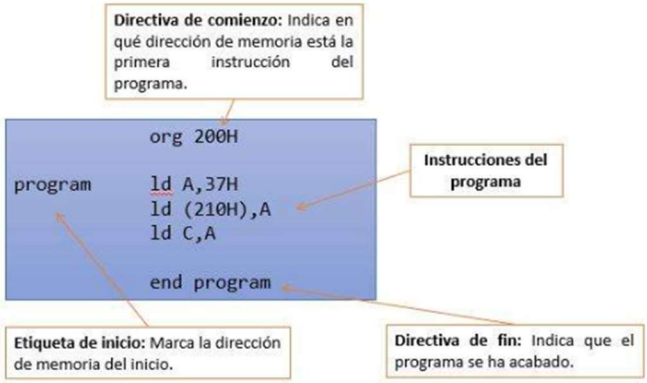
Arquitectura

La arquitectura del procesador Z80 se divide en varias partes importantes:

- Unidad de Control: Es la encargada de controlar el flujo de datos dentro del procesador y su sincronización.
- Unidad Aritmético-Lógica (ALU): Realiza operaciones aritméticas y lógicas en los datos que se encuentran en el registro acumulador.
- Registros: El procesador Z80 cuenta con varios registros, entre ellos se encuentran el registro acumulador, registros de propósito general, registros de índice, registros de segmento y registros de pila. Ya los hemos comentado con anterioridad.
- Unidad de Memoria: Esta unidad se encarga de leer y escribir en la memoria del sistema.
- Control de interrupciones: El Z80 puede manejar interrupciones de hardware y software. La unidad de control detecta las interrupciones y dirige la ejecución del programa hacia una rutina específica de manejo de interrupciones.
- Bus de direcciones: Se encarga de llevar la dirección de la memoria o dispositivo que se desea acceder.
- Bus de datos: Se utiliza para transferir datos entre el procesador y la memoria o dispositivos de entrada/salida.



5. Ensamblador Z80



Los números se pueden representar en diferentes bases en el ensamblador. Algunas instrucciones admiten diferentes bases, otras son dependientes de una base concreta.

- En decimal: 34D o, simplemente, 34. (Ojo 34D no funciona en ZEUS)
- En hexadecimal: 34H o, si comienza por letra, 0A4H
- En binario: 00110100B

Instrucción LD

La instrucción que más utilizaremos en nuestros programas en ensamblador será sin duda la operación de carga o instrucción LD. Sirve para:

- Meter un valor en un registro.
- Copiar el valor de un registro en otro registro.
- Escribir en memoria (en una dirección determinada) un valor.
- Escribir en memoria (en una dirección determinada) el contenido de un registro.
- Asignarle a un registro el contenido de una dirección de memoria.

La instrucción LD copia el valor contenido en el operando origen en el operando destino. Esta operación no es aritmética, ni lógica ni de desplazamiento, por lo que no afecta a ningún *flag* (F). Comparación de la instrucción LD del Z80, con las equivalentes del 8080 y 8086.

Datapoint 2200 & i8008	i8080	Z80	i8086/i8088
before ca. 1973	ca. 1974	1976	1978
LBC	MOV B,C	LD B,C	MOV BL,CL
--	LDAX B	LD A,(BC)	MOV AL,[BX]
LAM	MOV A,M	LD A,(HL)	MOV AL,[BP]
LBM	MOV B,M	LD B,(HL)	MOV BL,[BP]
--	STAX D	LD (DE),A	MOV [DX],AL [x]
LMA	MOV M,A	LD (HL),A	MOV [BP],AL
LMC	MOV M,C	LD (HL),C	MOV [BP],CL
LDI 56	MVI D,56	LD D,56	MOV DL,56
LMI 56	MVI M,56	LD (HL),56	MOV byte ptr [BP],56
--	LDA 1234	LD A,(1234)	MOV AL,[1234]
--	STA 1234	LD (1234),A	MOV [1234],AL
--	--	LD B,(IX+56)	MOV BL,[SI+56]
--	--	LD (IX+56),C	MOV [SI+56],CL
--	--	LD (IY+56),78	MOV byte ptr [DI+56],78
--	LXI B,1234	LD BC,1234	MOV BX,1234
--	LXI H,1234	LD HL,1234	MOV BP,1234
--	SHLD 1234	LD (1234),HL	MOV [1234],BP
--	LHLD 1234	LD HL,(1234)	MOV BP,[1234]
--	--	LD BC,(1234)	MOV BX,[1234]
--	--	LD IX,(1234)	MOV SI,[1234]

Ejemplos

- Cargar registro: origen = memoria -> destino = registro LD A, (1000)
- Almacenar registro: origen = registro -> destino = memoria LD (1000), A
- Transferir: origen = registro -> destino = registro LD A, B

Módulos de Direccionamiento

El direccionamiento en el contexto del ensamblador del Z80 es crucial para indicar dónde están los datos con los que se operará. Aquí está una explicación más detallada de los diferentes modos de direccionamiento que mencionaste:

- Direccionamiento extendido:
  Se utiliza cuando el operando es la dirección de memoria donde está el dato.
  Las operaciones con memoria solo pueden tener origen o destino en el registro A. Si se necesita cargar el dato en otro registro, se debe hacer en dos pasos.
  El operando consiste en 2 bytes para contener una dirección completa de memoria del Z80. Los registros dobles (BC, DE y HL) también pueden interactuar con la memoria.
- Direccionamiento indirecto:
  El operando es un par de registros que contienen la dirección de memoria donde está el dato.
  Los registros (A, B, C, D, E, H, y L) pueden aceptar como dirección de origen la contenida en el registro HL. El registro A también puede aceptar como origen los registros BC y DE. Todos estos registros también aceptan IX e IY.
- Direccionamiento indexado:
  El operando es un registro de 2 bytes (IX e IY) más un desplazamiento.
  La dirección contenida en el registro más el desplazamiento resulta en la dirección de memoria donde está el dato.
  Todos los registros de 8 bits (A..L) pueden ser destino en este modo.
- Direccionamiento a través de registro:
  El operando es un registro, y no se accede a memoria ya que el dato está contenido en el registro. Todos los registros de 8 bits (A...L) pueden ser origen y destino.
- Direccionamiento implícito:
  No hay operandos explícitos en la instrucción, ya que la instrucción actúa automáticamente sobre un registro concreto.
  No se accede a memoria ya que el dato está contenido en el registro especificado.
- Direccionamiento inmediato:
  El operando es el dato mismo, no se accede a memoria ni a registros adicionales.
- Direccionamiento inmediato extendido:
  Similar al direccionamiento inmediato, pero para números que se van a almacenar en registros de 2 bytes, como BC, DE, HL, IX, IY.

Formas de LD según el Direccionamiento

Atendiendo a los modos de direccionamiento, la instrucción LD se puede encontrar de las siguientes formas:

- LD A, (1000H); sólo registro A
- LD A, (BC) ; DE y HL también son válidos
- LD R, (HL) ; donde R = A, B, C, D, E, H, L
- LD R, (IX + d) ; donde d = desplazamiento
- LD R, (IY + d); ld HL, (IX+d o IY +d) da error, en cambio con BC y DE no da error
- LD R, n ; donde n es un número desde 00 a FF y R = A, B, C, D, E, H, L. R puede sustituirse por (HL), (IX + d) e (IY + d)

Viceversa para almacenar datos del registro en lugar de cargar datos en el registro.

- LD (1000H), A; sólo registro A

También se puede transferir el contenido de registro a registro:

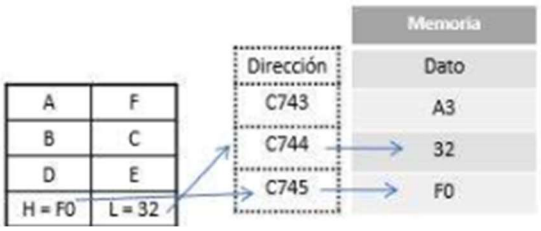
- LD R1, R2 ; donde R1 = A, B, C, D, E, H, L y R2 = A, B, C, D, E, H, L

LD también admite direccionar datos de 2 bytes:

- LD HL, (0C744H); BC, DE, IX, IY y SP también son válidos.

El byte menos significativo está almacenado en la posición indicada. El byte más significativo en la posición siguiente. Esto se llama Low Endian. Viceversa para almacenar datos de un par de registros en dos direcciones consecutivas de memoria.

LD (0C744H), HL ; BC, DE, IX, IY y SP también son válidos.



EX y EXX intercambio

En lugar de cargar o almacenar, es posible intercambiar valores entre pares de registros con la instrucción EX:

- EX DE, HL
- EX AF, AF'
- EX (SP), HL
- EX (SP), IX
- EX (SP), IY

Para intercambiar los valores de los registros actuales por los del segundo juego de registros:

- EXX