Trabajo Práctico Unidad 7:

# Diseño de una Computadora Digital

*Arquitectura de Computadoras*

UTN.BA

## CPU Y BUSES

1. ¿Qué es la CPU y cómo es su diagrama en bloques interno? Describir el funcionamiento de cada uno de sus módulos.
2. Enumerar sus registros y explicar brevemente para qué se usa cada uno.
3. ¿Qué es un Bus? Enumerarlos y describir cada uno.

## INSTRUCCIONES

1. ¿Qué es una instrucción?
2. ¿Qué es el código de una instrucción u “OpCode”?
3. ¿Qué es el formato de una instrucción?
4. ¿Cuál es el registro que almacena el OpCode y el campo Data de las instrucciones?
5. ¿Cuál es el módulo de la CPU encargado de interpretar las instrucciones?
6. ¿Qué es el ciclo de instrucción?
7. ¿Cómo se llaman las señales generadas por la UC para comandar la operación?
8. Suponiendo que el campo OpCode del IR consta de 4 bits y que dichos bits son las entradas de un decodificador, ¿cuántas instrucciones, como máximo, conforman el repertorio de instrucciones de dicho microprocesador?

## DIRECCIONAMIENTO

12. Suponiendo que se tiene una memoria de 4 posiciones en total:

1. ¿Cuántos bits se necesitan, como mínimo, para poder acceder a cada una de dichas posiciones? (Esto se conoce como “direccionar” la memoria) .

|  |  |
| --- | --- |
| **Posición** | **Memoria** |
| 0 | <Posición 0> |
| 1 | <Posición 1> |
| 2 | <Posición 2> |
| 3 | <Posición 3> |

1. Suponer ahora que se tiene una memoria de 16 posiciones en total.

¿Cuántos bits se necesitan para acceder a cada posición?

1. Si se tiene una memoria de 1024 posiciones, ¿Cuántos bits se necesitarán para direccionarla?
2. Tratar de encontrar una regla general en términos de potencias de dos.
3. Sabiendo que:

|  |  |  |
| --- | --- | --- |
| 210​ | 1024 B | 1 KB |
| 220​ | 1024 KB | 1MB |
| 230​ | 1024 MB | 1GB |

* 1. ¿Cuántos bits (o líneas del bus de direcciones) se necesitan para direccionar una memoria de 4MB?
  2. Idem punto **i**​ ​, si la memoria es de 16GB.
  3. ¿Cuál es el registro de la CPU que se utiliza para direccionar la memoria?
  4. ¿Qué relación debe haber entre dicho registro y el tamaño de la memoria? ¿Es importante para su dimensionamiento la longitud de la palabra de memoria?
  5. ¿Cuántos “cables” o “líneas” debería tener el bus de direcciones? ¿De qué depende?

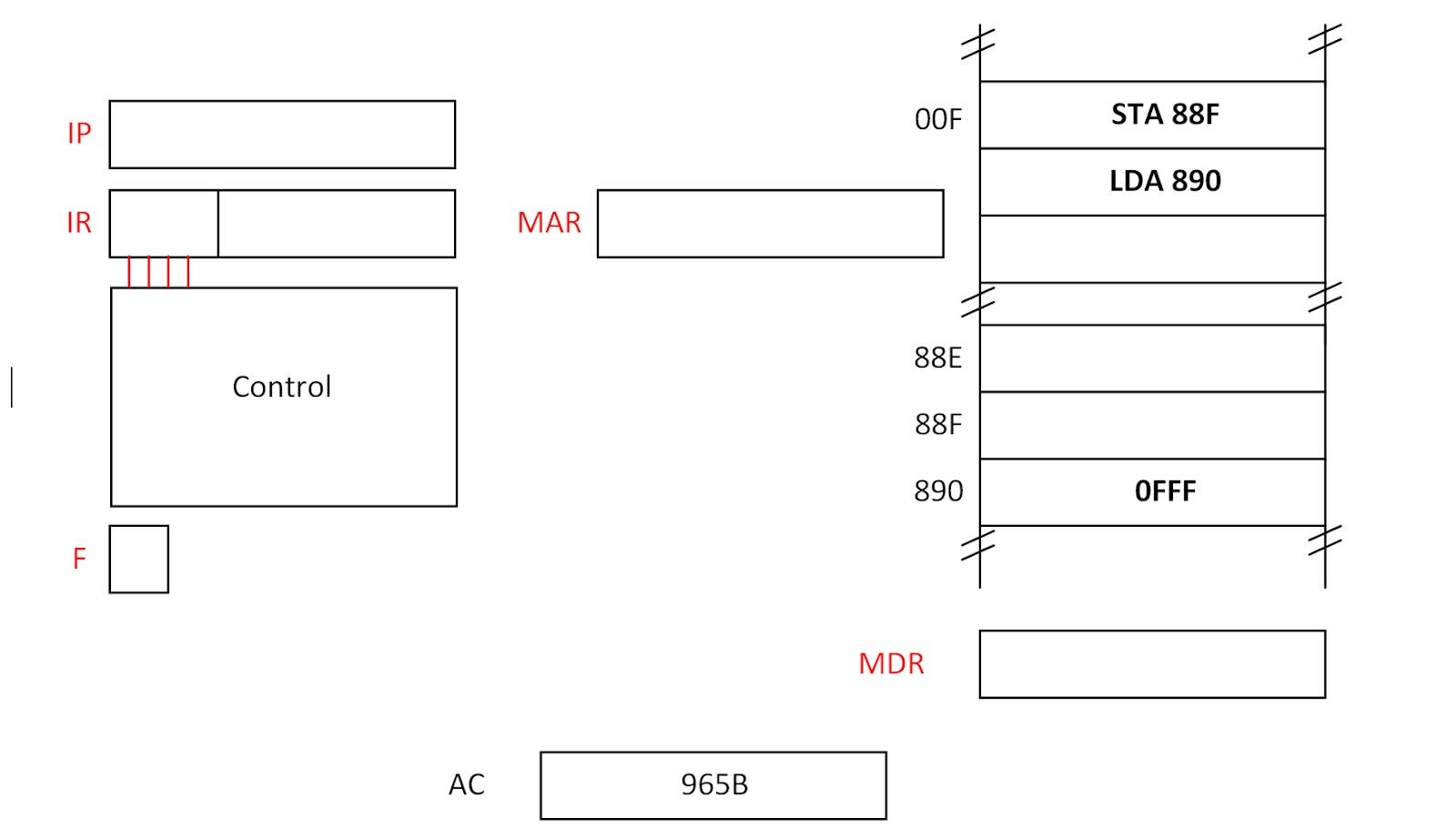
## FASES FETCH Y EXCECUTE

1. ¿Cuál es la función de la unidad de Control? Explicar sus fases
2. ¿Qué significa que la fase fetch depende del tiempo y del flag de control de fase?
3. Sabiendo que la fase fetch consta de 5 microoperaciones, a saber:

|  |  |
| --- | --- |
| f​1 = F.t​ ​0 | IP →MAR |
| f​2 = F.t​ ​1 | Contenido de palabra de memoria → MDR |
| f​3 = F.t​ ​5 | MDR → IR |
| f​4 = F.t​ ​1 | IP + 1 → IP |
| f​5 = F.t​ ​7 | 0 → F |

Implementar esta lógica con una memoria PLA *(*​ *ayuda: el estado del flag F y los t*​*ison las entradas y las f*​*i*​ *las salidas)*

1. ¿Qué es la fase **execute**​ ?​
2. Suponiendo que una CPU terminó la fase fetch de la instrucción **STA**​  **88F**,​ que está en la posición de memoria **00**​**F** y cuyo OpCode es ​​**3**:​
   1. Completar el gráfico con los datos de los registros, mostrando cómo se vinculan entre sí uniendo con flechas y explicar cómo es la secuencia de la generación de microinstrucciones de la fase execute:



010

f19

f18

f17

f16

965B

965B

88F

0

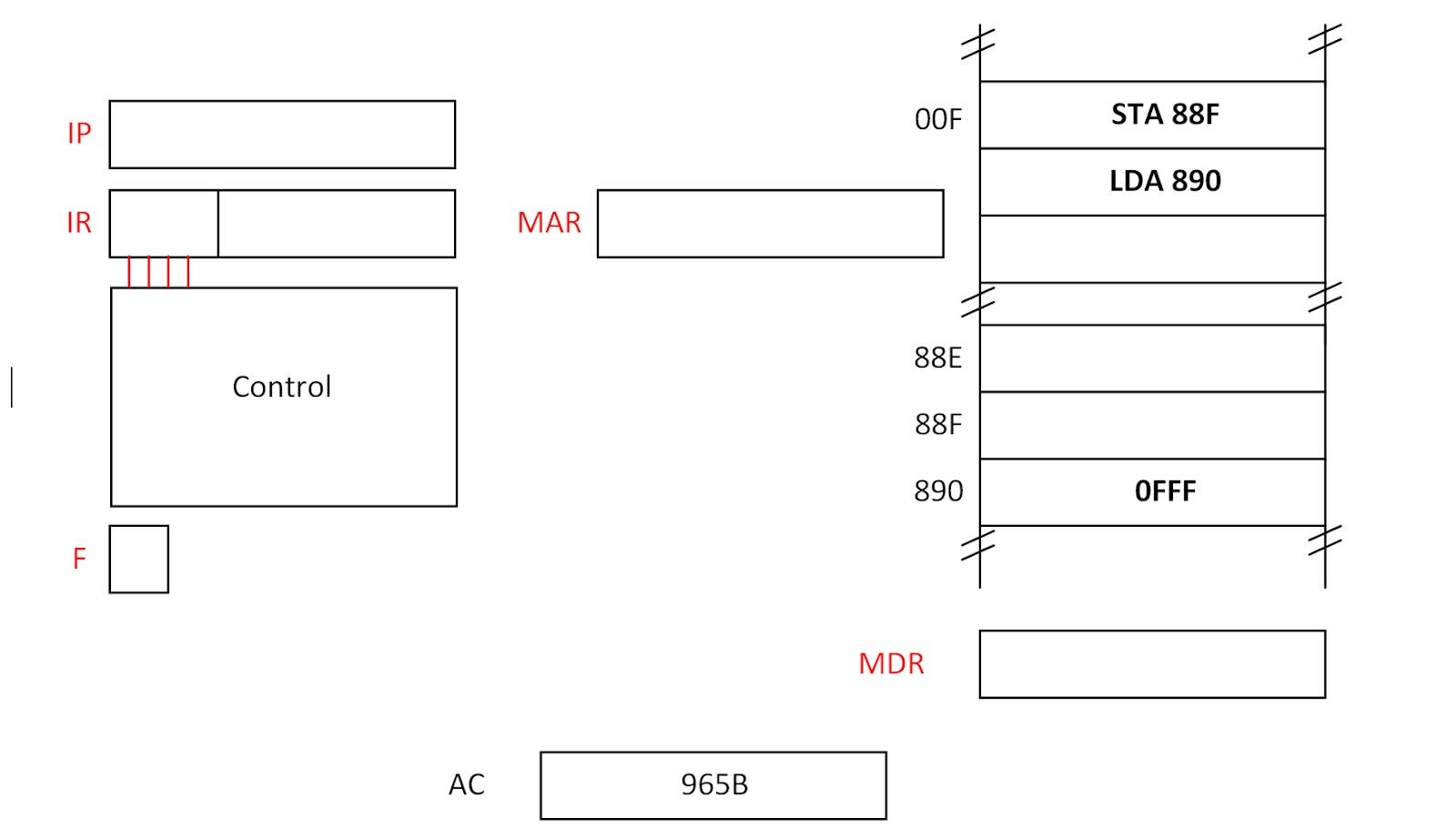
3

* 1. Escribir el valor binario de los bits que van del IR al control. Sabiendo que los mismos son la entrada a un decodificador *4*​​*a* ​*2*​*4*​, ¿Qué salida de dicho decodificador será la activa?
  2. Escribir las funciones de las señales de microoperaciones de la ejecución de esta instrucción.
  3. Realizar la implementación de la lógica del punto **c**​ con una memoria PLA

(sólo para este código de operación)

* 1. En base al gráfico, ¿cuál es la longitud de la palabra de memoria?
  2. ¿Cuántos “cables” o “vías” debería tener entonces el bus de datos? ¿De qué depende?

1. Una vez completada la ejecución de la instrucción del punto 17, ¿cómo sigue la ejecución del programa? Repetir el ejercicio 17 (puntos a. b. c. y d.) para la próxima instrucción *(*​ *ayuda: mirar el IP)* 
   1. Completar el gráfico con los datos de los registros, mostrando cómo se vinculan entre sí uniendo con flechas y explicar cómo es la secuencia de la generación de microinstrucciones de la fase execute:



0FFF

011

f9

f7

F8

f6

965B

0FFF

890

0

1

* 1. Escribir el valor binario de los bits que van del IR al control. Sabiendo que los mismos son la entrada a un decodificador *4*​​*a* ​*2*​*4*​, ¿Qué salida de dicho decodificador será la activa?
  2. Escribir las funciones de las señales de microoperaciones de la ejecución de esta instrucción. COP1 = Y1
  3. Realizar la implementación de la lógica del punto **c**​ con una memoria PLA

(sólo para este código de operación)

1

1. Si la siguiente instrucción es **INC**​ ,​ ¿Cuál será el valor del acumulador luego de la operación? ¿Y en qué estado quedarán seteados los flags del Status Register?

## SET DE INSTRUCCIONES

1. ¿Qué es un ​**mnemónico**​? ¿Para qué sirve?
2. Estudiar del libro el set completo de instrucciones presentado y explicar qué función realiza cada una de ellas:

|  |  |
| --- | --- |
| Instrucción | Descripción |
| LDA HHH | Transfieren el dato de la dirección HHH al acumulador |
| ADA HHH | Realiza la suma binaria entre el dato de la dirección HHH y el acumulador |
| SHR | Desplaza toda la información del acumulador un bit a la derecha, dividiendo todo por 2 |
| STA HHH | Almacena los datos del acumulador en la dirección HHH |
| HLT | Inactiva las señales de tiempo para que no se produzca la fase Fetch y finaliza el programa |
| JMP HHH | Realiza un salto incondicional en las instrucciones del programa a la dirección HHH, rompiendo el orden secuencial del programa transfiriendo esa dirección al IP |
| ANA HHH | Realiza el producto lógico entre los bits de la dirección HHH y los del acumulador |
| XOA HHH | Realiza la suma/disyunción exclusiva entre los bits de la dirección HHH y los del acumulador |
| CLA | Pone el acumulador en 0 |
| CMA | nvierte los bits del acumulador (Realiza el complemento restringido) |
| INC | Suma una unidad al acumulador. |
| SNA HHH | Realiza un salto condicional que permite tomar una decisión en base a una condición establecida, en este caso, que la flag S (Signo) sea 1, es decir, que el acumulador sea un número negativo. Si cumple la condición, se realizará el salto de instrucción a la dirección HHH, transfiriéndola al IP. Caso contrario, no se dará ningún salto. |
| SZA HHH | Realiza un salto condicional de la misma manera que SNA HHH, con la condición de que la flag Z (Cero) sea 1, es decir, que el acumulador sea 0. De ser así, realiza el salto, si no, no lo hace |
| SCA HHH | Realiza un salto condicional con la condición de que la flag C (Carry) sea 1, es decir, el último acarreo es 1 |
| INP | Permite la entrada de un dato desde un periférico de entrada, el cual ingresa a través de un registro ENT del periférico al acumulador. Se ejecuta a través de la flag FEN, que evalúa si el periférico tiene datos para transferir. Cuando los tiene, el valor de FEN será 1 y habrá información que transferir. Caso contrario, será 0 y significa que el periférico no envió datos al registro |
| OUT | Permite la salida de un dato almacenado en el acumulador hacia un dispositivo periférico. La transferencia se hace entre el acumulador y el dispositivo de salida. Se ejecuta cuando se hace efectiva la salida mediante la flag FSA. Si FSA es 0, el periférico no está listo y no puede enviar información al periférico. Cuando es 1, puede enviar la información, y vuelve a 0 para evitar seguir mandando información |

1. ¿Todas estas instrucciones necesitan leer/escribir un dato en memoria?
2. ¿Qué instrucción/es afectan directamente el IP?
3. ¿Cuáles instrucciones activan la ALU y el Status Register?

RELOJ

1. ¿Qué es un ciclo de clock?
2. ¿Qué es un ciclo de máquina?
3. Si una CPU tiene un reloj de 1GHz, ¿cuál es el tiempo “t​i” de cada pulso de clock?​
4. ¿Cuánto tiempo es un ciclo de máquina para la CPU del punto anterior, si el ciclo de máquina está conformado por 8 ciclos de clock?

1,2,3) La unidad central de procesamiento (conocida por las siglas CPU ) es el *hardware* dentro de una computadora u otros dispositivos programables.

Su diagrama en bloques es:

Dispositivos de entrada/salida o periféricos:

Por medio de los mismos se ingresan los datos e instrucciones al computador y se obtienen los resultados del proceso.

Unidad Central de Procesos o Central Processing Unit: (CPU)

Es la encargada de realizar el proceso con los datos e instrucciones.  
En la actualidad la CPU viene integrada en un solo circuito conocido con el nombre de microprocesador.  
En definitiva, un microprocesador es un circuito integrado capaz de ejecutar programas y controlar las unidades necesarias, para dicha ejecución.  
Dispone de dos bloques principales:

a) Unidad Aritmética y Lógica o Arithmetic Logic Unit: (ALU)

Es donde se realizan las operaciones de los datos.

b) Unidad de Control:

Tiene como misión supervisar todo el proceso, para lo cual recibe una señal eléctrica de sincronismo, de un circuito llamado reloj o clock.

Memoria:

Es cualquier lugar capaz de contener datos, programas y/o resultados de procesos.  
Se las puede clasificar de la siguiente manera:

a) Memoria Central:

Todo computador viene provisto de fábrica con una memoria central, también llamada memoria principal o memoria interna, que está constituida por las memorias RAM y ROM, las cuales cumplen funciones diferentes:

a1) RAM (Random Access Memory)

Es la memoria que contiene de forma temporaria el programa, los datos y los resultados que están siendo utilizados por el usuario del computador.  
Este tipo de memoria es volátil, es decir, pierde su contenido cuando se apaga el computador.

a2) ROM (Read Only Memory)

Esta memoria viene grabada de fábrica con una serie de programas que son indispensables para el funcionamiento del computador, por lo que solamente puede ser leída por el usuario y no escrita. Este tipo de memoria no es volátil.

b) Memoria auxiliar:

La memoria auxiliar, también llamada memoria secundaria, se usa para almacenar datos, información y cada uno de los programas que se van a necesitar en el computador. Puede ser magnética (como el caso del disco duro) u óptica (como el caso del CD-ROM).

Buses:

Todas las partes del computador anteriormente citadas deberán estar relacionadas entre sí; para ello, el mismo dispone de vías de comunicación llamadas comúnmente bus. Un bus necesita tener perfectamente definidas sus características en los siguientes aspectos:

- conexión mecánica: son los conectores (o "enchufes") utilizados;  
- conexión eléctrica: son las señales eléctricas utilizadas (significado, valores de las tensiones, tiempos de establecimiento, etc.);  
- protocolo de comunicación: son las reglas que deben seguirse para establecer una comu-nicación.

De esta forma, cualquier dispositivo de entrada/salida cuyo bus se adapte a las especificaciones mecánicas, eléctricas y al protocolo de comunicación de la CPU, podrá conectarse a la misma para establecer una comunicación.  
En el computador se distinguen:

Bus de Direcciones:

Es un conjunto de líneas unidireccionales que salen de la CPU, y seleccionan los dispositivos de entrada/salida o la posición de la memoria con la que va a trabajar.

Bus de Datos:

Es un conjunto de líneas bidireccionales para el intercambio de datos, instrucciones y resultados entre la CPU y los dispositivos de entrada/salida o la memoria, (que previamente han sido seleccionados por el bus de direcciones).

Bus de Control:

Es un conjunto de líneas de entrada y/o salida de la CPU, que permiten coordinar todas las operaciones del computador (no está dibujado en el esquema, ya que serían varias líneas uniendo todas las partes del computador).

Diagrama

Descripción generada automáticamente

4) Es una operación elemental que el procesador puede cumplir. Las instrucciones se almacenan en la memoria principal, esperando ser tratadas por el procesador.

5) un opcode (operation code) o código de operación, es la porción de una instrucción de lenguaje de máquina que especifica la operación a ser realizada. Su especificación y formato serán determinados por la arquitectura del conjunto de instrucciones (ISA) del componente de hardware de computador - normalmente un CPU, pero posiblemente una unidad más especializada

6) Los formatos de esta serie de computadores se caracterizan por tener código de operación de longitud fija y longitud total de instrucción variable. No es ortogonal y los modos de direccionamiento se codifican en el código de operación. La longitud de un formato de instrucción puede ser de 16 bits, 32 bits o 48 bits

7) Los operandos sobre los cuales los opcodes aplican pueden, dependiendo de la arquitectura del CPU, consistir de. registros, valores en memoria, valores almacenados en la pila, puertos de I/O, bus, etc. Las operaciones que un opcode puede especificar pueden incluir aritmética, copia de datos, operaciones lógicas, y control del programa.

8) La unidad de control es uno de los tres bloques funcionales principales en los que se divide una CPU . Los otros dos bloques son la unidad de proceso y el bus de entrada/salida.

Su función es buscar las instrucciones en la memoria principal, decodificarlas (interpretación) y ejecutarlas, empleando para ello la unidad de proceso.

9) Un ciclo de instrucción es el período que tarda la unidad central de proceso ([CPU](https://es.wikipedia.org/wiki/CPU)) en ejecutar una instrucción de [lenguaje máquina](https://es.wikipedia.org/wiki/Lenguaje_m%C3%A1quina)

10) Existen dos tipos de unidades de control: las cableadas, usadas generalmente en máquinas sencillas, y las microprogramadas propias de máquinas más complejas. En el primer caso, los componentes principales son el circuito de [lógica secuencial](https://es.wikipedia.org/wiki/L%C3%B3gica_secuencial), el de control de estado, el de [lógica combinacional](https://es.wikipedia.org/wiki/L%C3%B3gica_combinacional) y el de emisión de reconocimiento de señales de control. En el segundo caso, la microprogramación de la UC se encuentra almacenada en una micromemoria, a la cual se accede de manera secuencial para posteriormente ir ejecutando cada una de las microinstrucciones.

11) Un decodificador de 4 bits o entradas tendrá en total 16 salidas. Por lo tanto, si un OpCode del IR tiene 4 bits y estos son las entradas de un decodificador, el repertorio de instrucciones del microprocesador tendrá un total de 16 instrucciones como máximo

12) a- Para 4 posiciones de memoria, hacen falta 2 bits.

B - Para 16 posiciones de memoria, se necesitan 4 bits.

C - Para 1024 posiciones de memoria, son necesarios 10 bits

1. Si para 1 MB necesito 10 bits, porque 210 = 1024 KB = 1 MB, entonces para 4 MB necesito 12 bits, porque 212 = 22.210 = 4 MB
2. Si para 1 GB necesito 30 bits, porque 230 = 1024 MB = 1 GB, entonces para 16 GB necesito 34 bits, porque 234 = 24.230 = 16 GB.
3. El registro de la CPU es el Registro de Acceso a Memoria
4. Dicho registro debe tener la misma cantidad de bits que la cantidad total de bits que determinan el tamaño de la memoria. Si con longitud de la palabra de memoria se refiere al tamaño de los datos dentro de una dirección, entonces no es demasiado importante para su dimensionamiento la longitud de la palabra de memoria, ya que para el MAR lo que importa es la dirección en la memoria, no la información dentro de ella.
5. El Bus de Direcciones debería tener la misma cantidad de líneas que el Registro de Acceso a Memoria, por lo que depende de la cantidad total de direcciones de la memoria

13) La función de la Unidad de Control es la de tomar instrucciones del programa almacenado en memoria y ejecutarlas. Se divide en dos fases: Fetch y Execute. En la primera fase, la Unidad de Control busca las instrucciones en memoria a través de su dirección y los datos en ella, capta la siguiente instrucción, y prepara a la instrucción actual en un registro de instrucciones para que sea decodificada. Una vez decodificada, comienza la fase Execute, donde se ejecutan y se operan las instrucciones

14) Significa que las microoperaciones de la fase Fetch están sincronizadas a través del tiempo y que sólo puede ocurrir siempre y cuando el flag de control de fase esté en 1. De no depender del tiempo, las instrucciones ocurrirían en cualquier momento y en desorden, y si no dependiera del flag de control, entonces no habría un límite entre las fases Fetch y Execute

16) Es aquella en la cual, una vez captadas las instrucciones por la fase Fetch, se encarga de ejecutar instrucciones, acorde a la secuencia de microoperaciones que tenga el código de la instrucción definidas en la Unidad de Control

17) a- Como se ve en el gráfico, el registro de instrucción indica la dirección en memoria donde está la palabra cuyos datos serán transferidos al MAR. Luego, debido a la instrucción STA, el Control interpreta que el dato que está en el acumulador debe transferirse a la posición de memoria 88F. Para ello, transfiere el dato del acumulador a la MDR. Por último, la Unidad de Control escribe los datos de MDR en la dirección en memoria que le asigna el MAR. Todo esto ocurre siempre que la flag de Fetch sea 0. De otra manera, estaría en fase Fetch y no en Execute. Cuando la flag de Fetch sea 1, empezará con la siguiente instrucción, la que indica el IP.

b- Los bits que van del IR al control son los que determina el OpCode, es decir, 3h, que en binario es 0011b. Si son la entrada de un decodificador 4 entradas y 16 salidas, la salida activada será la cuarta

C- f1 = (~F).t0.COP3 f2 = (~F).t1.COP3 f3 = (~F).t3.COP3 f4 = (~F).t7.COP3 DATA => MAR AC => MDR MDR => WORD 1 => F

E- En base al gráfico de la instrucción del punto A, los datos contenidos en las direcciones de memoria constan de 4 dígitos hexadecimales, es decir, 16 bits, por lo que la longitud d la palabra de memoria es de 16 bits

F-  El bus de datos debería tener 16 vías. Depende de la longitud de palabra de memoria y de la cantidad de bits que puede almacenar el MDR.

18- A) Como se ve en el gráfico, el registro de instrucción indica la dirección en memoria donde está la palabra cuyos datos serán transferidos al MAR. Luego, debido a la instrucción LDA, la Unidad de Control debe cargar los datos de la dirección de MAR en el acumulador, por lo que se buscan los datos en la dirección de MAR y se los transfieren a la MDR, los cuales lo transferirán al acumulador. Todo esto mientras la flag F sea 0. B) Los bits que van del IR al control son los que determina el OpCode, es decir, 4h, que en binario es 0100b. Si son la entrada de un decodificador 4 entradas y 16 salidas, la salida activada será la quinta. C) f1 = (~F).t0.COP4 f2 = (~F).t1.COP4 f3 = (~F).t5.COP4 f4 = (~F).t7.COP4 DATA => MAR WORD => MDR MDR => AC 1 => F

19- INC suma 1 al dato del acumulador, por lo que su valor será 1000. Si la suma es: 0FFF + 0001 en hexadecimal, en binario sería: Carry 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0FFF + 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0001 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1000 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 Por lo tanto, los flags del Status Register S (Signo), V (Overflow), C (Carry) y Z (Zero) quedan en cero

20- Un mnemónico son tres letras que definen un OpCode. Sirve para expresar programas sin necesidad de recordar cadenas de dígitos binarios asignados a las instrucciones

22- No todas. Las que no necesitan leer o escribir datos de memoria son aquellas que no especifican direcciones de memoria

23- Las instrucciones de salto, ya sea condicional o incondicional (JMP HHH, SNA HHH, SZA HHH y SCA HHH)

24- Las que realizan operaciones en el acumulador, tales como LDA HHH, ADA HHH, ANA HHH, XOA HHH, CLA, CMA, INC y SHR

25- Un ciclo de clock/reloj es el intervalo entre dos pulsos consecutivos de reloj generados por un oscilador y circuitos asociados que generan pulsos cuyos anchos y separación se miden con precisión

26-  Es el tiempo de una secuencia repetitiva formada por 4, 8 o 16 señales de tiempo.

27- frecuencia = Ciclos/Segundo 1 GHz = 109 Hz = 1.000.000.000 Hz = 1.000.000.000 ciclos/segundo. 1.000.000.000 ciclos ocurren en 1 segundo, 1 ciclo ocurre en 0,000000001 segundos, es decir, 1 nanosegundo. Un ciclo de reloj equivale a dos pulsos adyacentes de reloj, por lo que cada pulso dura 0,5 nanosegundos.