TRABAJO PRÁCTICO:

Unidad 7

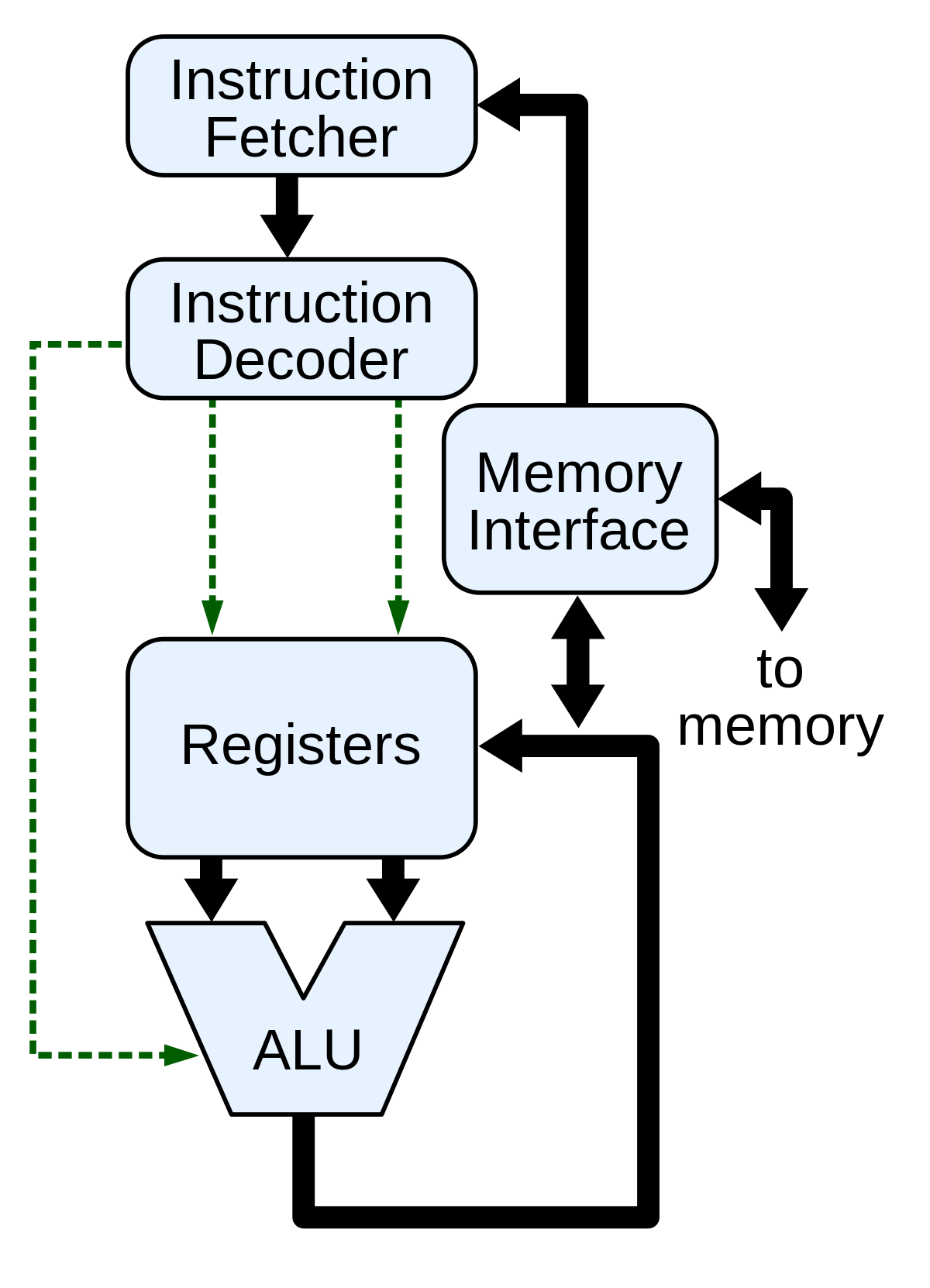
Arquitectura de Computadores

CURSO K1028

|  |  |  |
| --- | --- | --- |
| **Bencina**, Morena | 1772922 | mbencina  @est.frba.utn.edu.ar |
| **Colotto**, Nahuel | 1774189 | ncolotto  @est.frba.utn.edu.ar |
| **Enrique Zabala**, Laureano Gaspar | 1770640 | lenriquezabala  @est.frba.utn.edu.ar |
| ***x* Lopez Joffre**, Dolores | 1762023 | dlopezjoffre  @est.frba.utn.edu.ar |
| **Tossi**, Santiago Nicolas | 1773355 | stossi  @est.frba.utn.edu.ar |

# CPU y buses

1. La CPU, Central Processing Unit, es la unidad central de proceso y se encarga de:   
   Tratamiento de instrucciones, operación de los datos y cálculos de dirección de memoria.



La CU debe enviar a la memoria la dirección de la palabra donde se encuentra la instrucción, una orden de lectura y una orden de transferencia de la instrucción. La CU retiene la dirección de la instrucción en un registro especial denominado IP o Instruction Pointer. Una vez que la CU envía a memoria el contenido del IP, da la orden de lectura para que la palabra implicada, se almacene en el registro de palabra de memoria (MDR o Memory Data Register). Por último, la etapa fetch termina con la transferencia de la instrucción leída a un registro

interno de la CU, donde la instrucción permanece almacenada mientras dure su ejecución (denominado registro de instrucción, **IR**).

La ALU es la encargada de operar aritmética y lógicamente datos, los cuales son almacenados en registros

1. Los registros más cruciales en el ciclo *fetch-execute* son:

* Registro de Fase (F): Este registro almacena 1 si el procesamiento se encuentra en la fase de búsqueda (o *fetch*) o 0 si está en la fase de ejecución (o *execute*).
* Instruction Pointer (IP):
* Instruction Register (IR): Este registro contiene el código de instrucción y su operación correspondiente (si este existiese) a ejecutar.
* Accumulator (AC): Un registro en el que se almacenan temporalmente los resultados aritméticos y lógicos intermedios, tratados por la ALU.
* Status Register (SR o SZVC): Es el registro que almacena 4 *flags* que marcan el estado en el que se encuentra el último valor almacenado en el registro AC. Estas flags son S (de signo, marca 1 si el último valor es negativo), Z (de zero, marca 1 si el último valor es 0), V (de *overflow*, marca 1 si el último valor sobrepasó la cantidad de bits máxima del registro) y C (de *carry*, que marca si hubo transferencia de un valor a una posición más significativa).
* Memory Address Register (MAR):
* Memory Data Register (MDR):

1. Los buses son puentes metálicos que transportan información de un lugar a otro en una computadora; normalmente conectan la CPU con la memoria principal y los registros intermedios. Los buses pueden ser:

* **Buses de datos**: permiten la transferencia de grupos de bits desde o hacia espacios de memoria.
* **Buses de control**: se envían órdenes de Lectura (RD) o Escritura (WR) hacia la Memoria Principal.
* **Buses de dirección:** Comunica a la MMU con la memoria principal.

# **I**nstrucciones

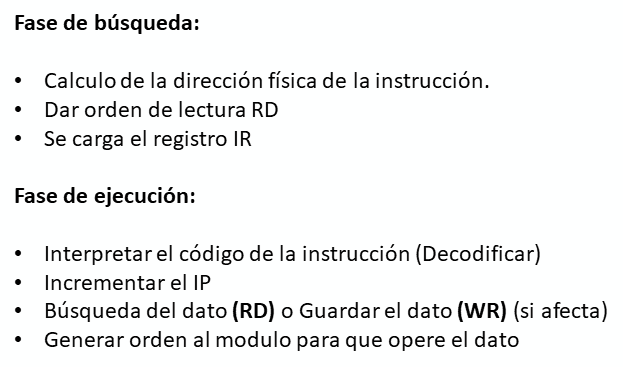
1. Una instrucción es un conjunto de datos insertados en una secuencia estructurada o específica que el procesador interpreta y ejecuta. Hablando más específicamente de las microoperaciones de un procesador, podemos referirnos a conjuntos de microoperaciones que interactúan estrechamente con la arquitectura de una computadora, modificando sus registros al escribirlos y leerlos con la intención de realizar una tarea específica.
2. El código de una instrucción es la combinación de bits que la unidad de control de la CPU interpreta para generar las microoperaciones que permitan su ejecución.
3. Se lo denomina al formato de la instrucción como la forma de agrupar estos bits en entidades diferenciadas determina la estructura de la instrucción.
4. El registro que almacena el opcode y el campo data de las instrucciones es **IR.**
5. **CU** es el módulo de la CPU encargado de interpretar las instrucciones.
6. El ciclo de instrucción es el período que tarda la CPU en ejecutar una instrucción de lenguaje de máquina.
7. La CU debe enviar a la memoria la dirección de la palabra donde se encuentra la instrucción, una orden de lectura y una orden de transferencia de la instrucción.
8. El repertorio de instrucciones del microprocesador conformarán como máximo 16 instrucciones.

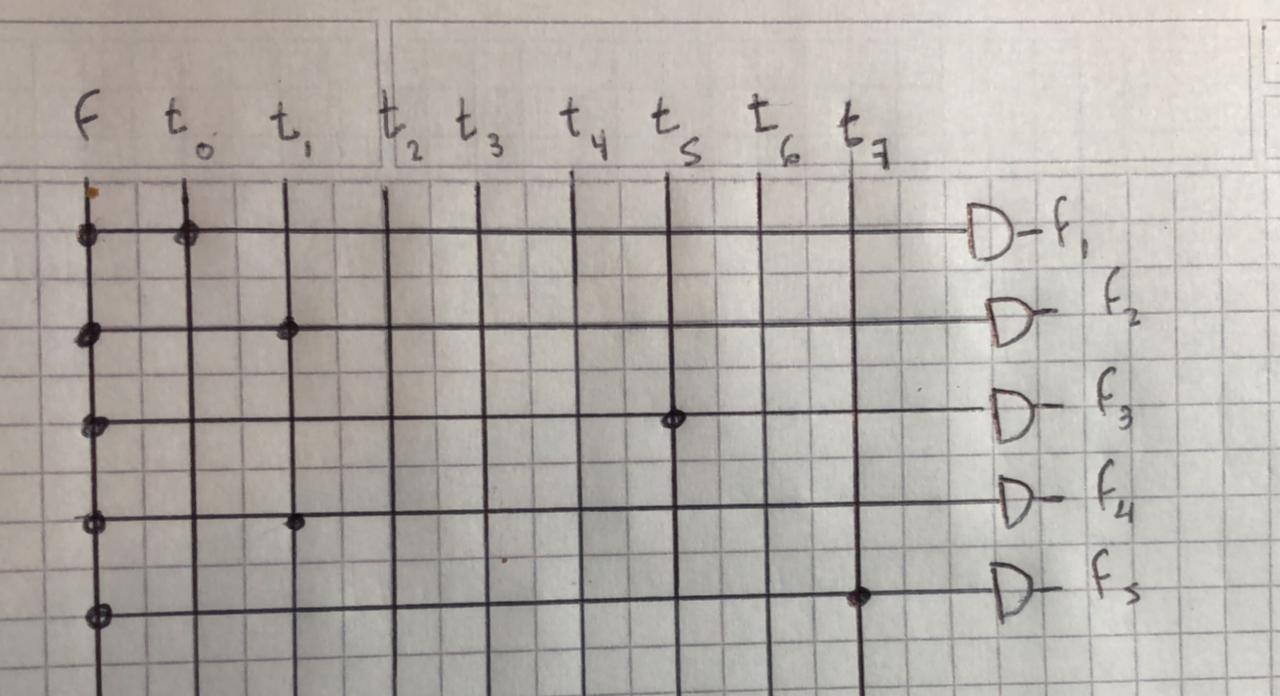
# Direccionamiento

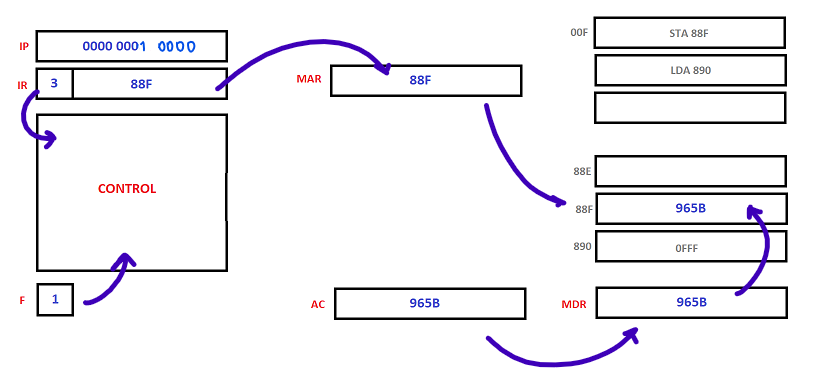
* 1. Se necesitan 2 bits.
  2. 4 bits
  3. 10 bits
  4. 2^N donde N es la cantidad de bits para direccionar y 2^N es la cantidad total de direcciones.
     1. 4 MB= 2^2\* 2^20= 2^22, se necesitan 22 bits.
     2. 16GB= 2^4 \* 2^30= 2^34, se necesitan 34 bits
     3. El registro MAR (Memory Address Register)
     4. la relación entre la MAR y el tamaño de la memoria, es que el tamaño de la MAR dependerá de la cantidad de posiciones que debe redireccionar en la memoria. No es importante para su dimensionamiento la longitud de cada palabra de la memoria, ya que esta solo se ocupa de redireccionar y no tiene en cuenta su longitud.  
        Donde N bits de MAR, permitirán redireccionar 2^N posiciones en la memoria.
     5. El bus de direcciones tendrá N cantidad cables ingresando a la MAR en una memoria de 2^N direcciones.

# Fases *fetch* y *execute*

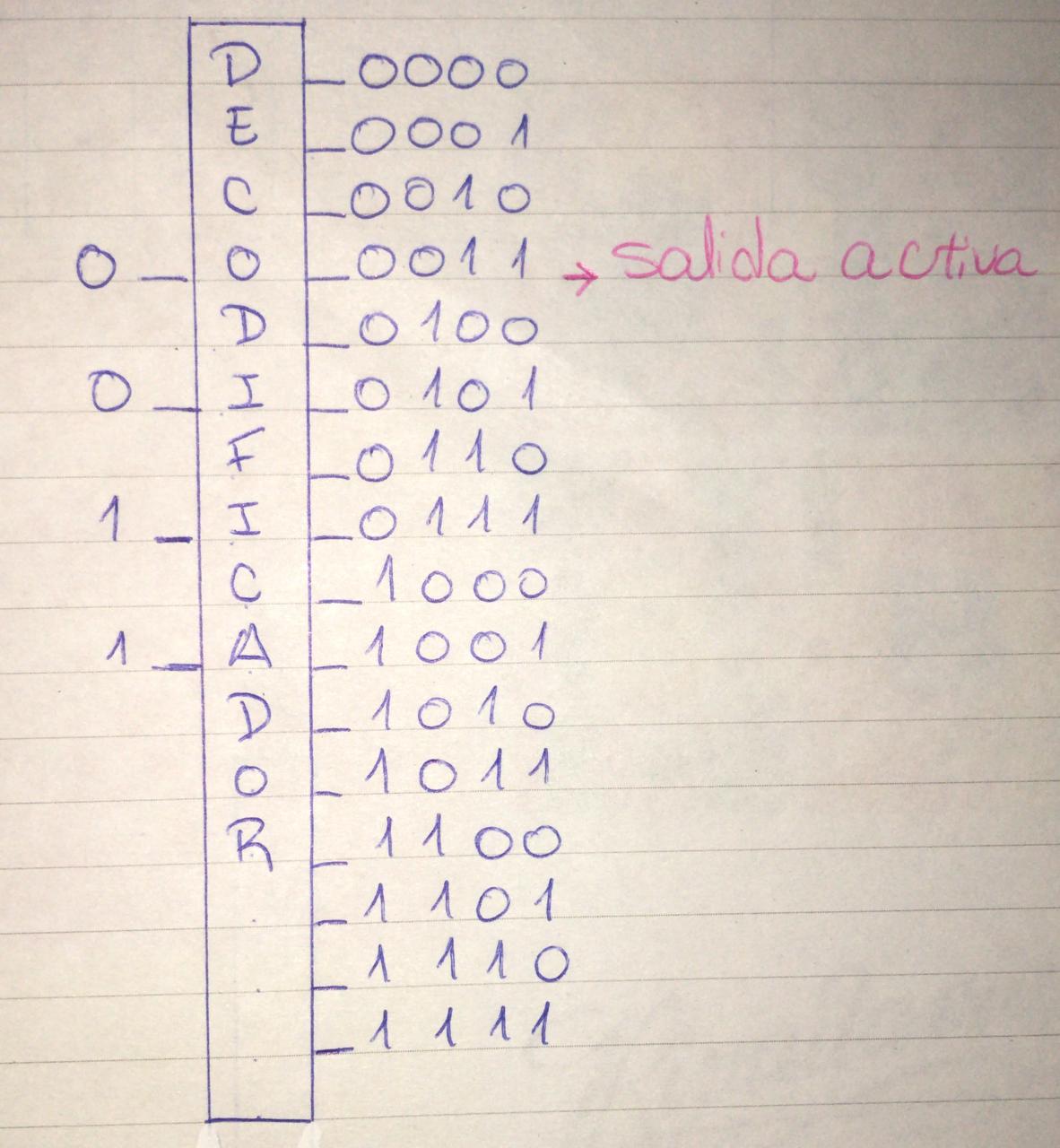
1. La unidad de control de la CPU dirige una secuencia de pasos de modo que la computadora lleve a cabo un ciclo completo de ejecución de una instrucción, y hacer esto con las instrucciones que conste el programa. Sus fases son:



1. La fase fetch depende del tiempo y del flag de control de fase ya que las microoperaciones que definen la fase fetch se producen cuando el flag F está en 1 en los pulsos de reloj.
2. 
3. La fase execute consta de 3 partes: Interpretación del código de instrucción, búsqueda del dato afectado (si afecta a dato) por la instrucción y la generación de órdenes al módulo que opera sobre ese dato.
5. Primero, se transfiere la dirección donde se va a recopilar la información hacia el MAR. Se transfiere el contenido del acumulador al MDR, y luego se da la orden de escritura. El MDR transfiere su contenido a la dirección correspondiente. Por último, se carga un 1 en el registro F para iniciar la etapa fetch y buscar la próxima instrucción.





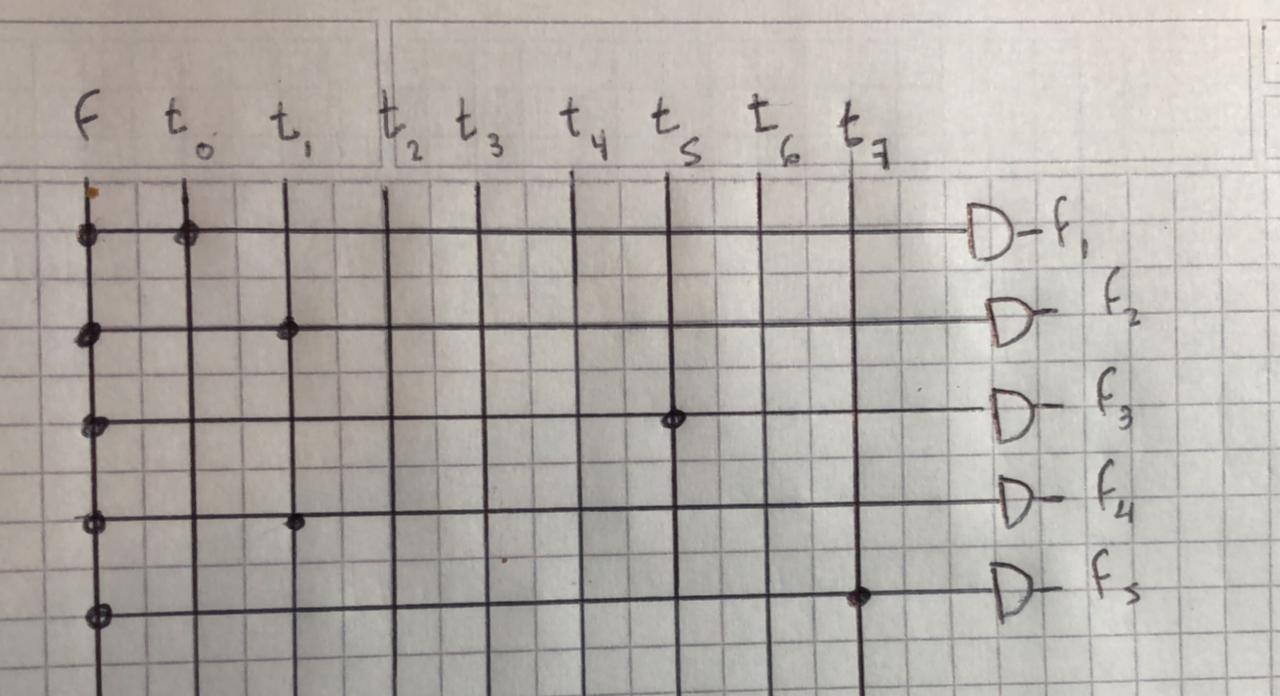


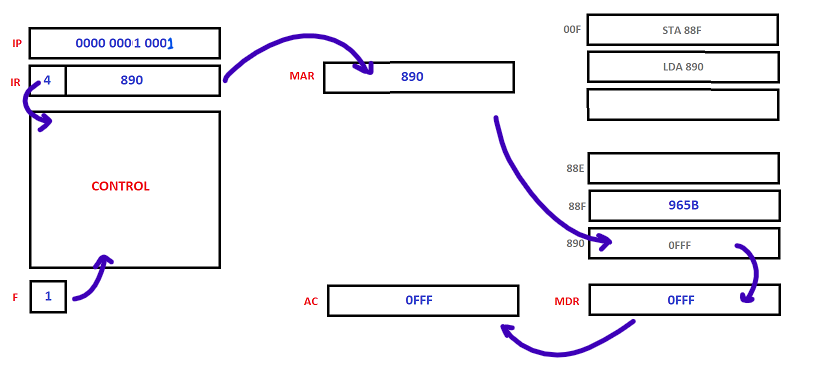
1. DATA→ MAR

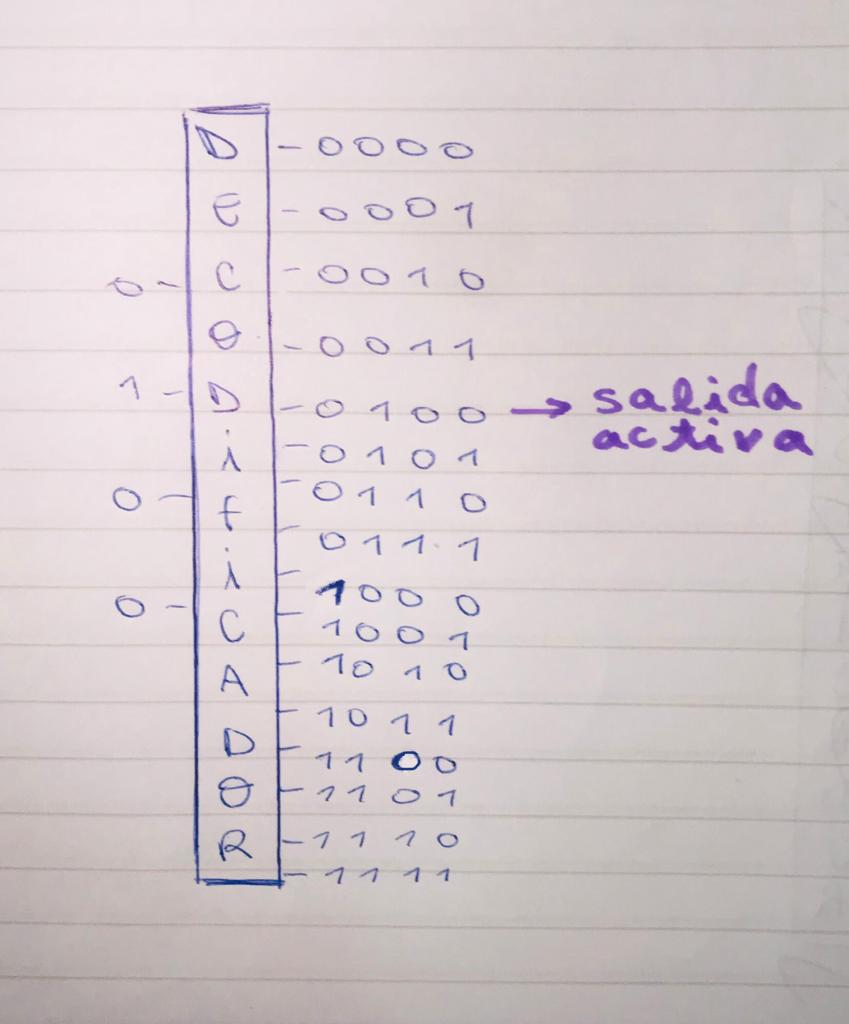
AC→ MDR

0→ ~WR

1→ F

1. 
2. 16 bits. Esta almacena instrucciones de 16 bits (4 bits de opcode + 12 bits de data) y además valores de cuatro dígitos hexadecimales, que pueden ser representados con 16 bits.
3. 16 bits, también. La longitud de los buses de datos que transportan el valor del MDR y el control dependen de la cantidad de bits del mismo MDR, que asimismo depende de la longitud de la palabra de memoria.
   1. Primero, se transfiere la dirección donde se va a almacenar la información hacia el MAR. Se da la orden de lectura, y luego se transfiere el contenido de la posición de memoria al MDR. El MDR transfiere su contenido a la dirección correspondiente. Por último, se carga un 1 en el registro F para iniciar la etapa fetch y buscar la próxima instrucción.

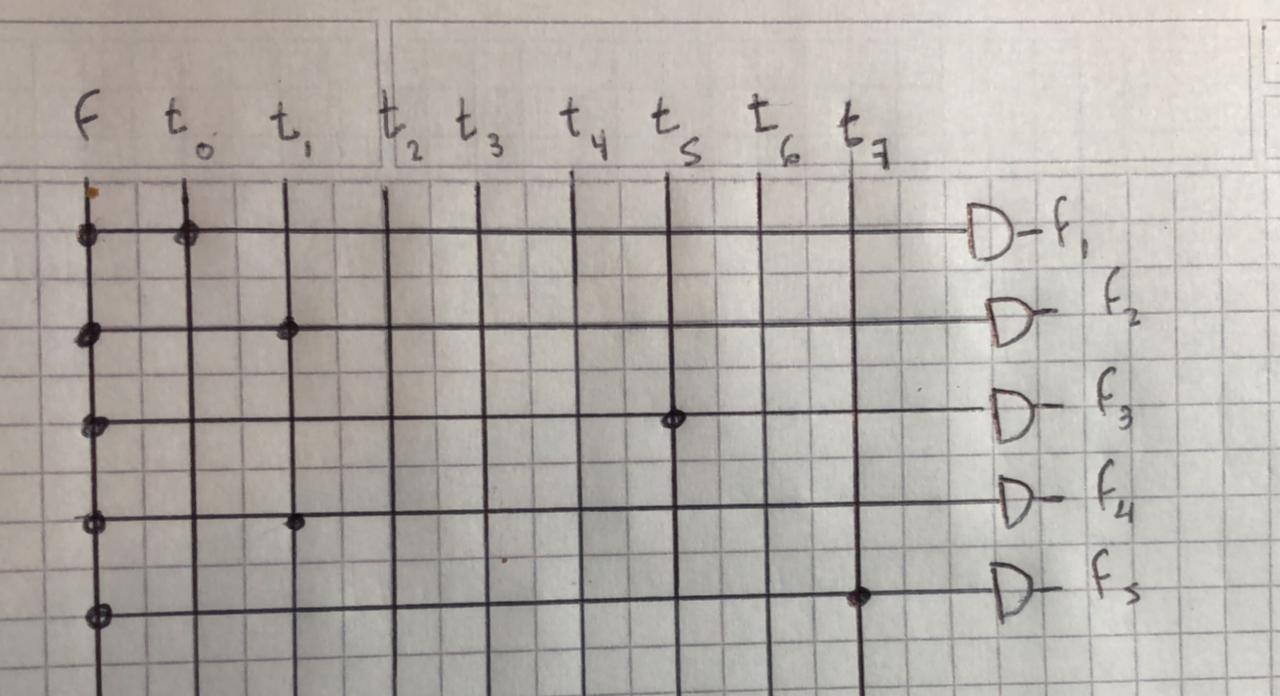


* 1. 
  2. DATA → MAR

1 → ~WR

MDR → AC

1 → F

* 1. 

1. La instrucción INC incrementa el valor del acumulador por 1. El acumulador pasaría a almacenar el valor 1000 (0FFF+1) y las flags tomarían los estados S=0, Z=0, V=0, C=1.

# Set de instrucciones

1. Un mnemónico es un conjunto de 3 símbolos en hexadecimal que se utilizan para definir el opcode y la dirección del operando, es un lenguaje que permite expresar los programas al usuario de manera más fácil.

|  |  |
| --- | --- |
| ***Operación*** | ***Descripción*** |
| *LDA HHH* | Carga en la palabra HHH el valor del acumulador |
| *ADA HHH* | Suma al acumulador la palabra HHH |
| *STA HHH* | Almacena en el acumulador la palabra HHH |
| *SHR* | Desplaza hacia la derecha el valor del acumulador |
| *HLT* | Fin del programa |
| *ANA HHH* | Realiza un producto lógico entre el valor del acumulador y el dato obtenido |
| *XOA HHH* | Realiza Or Exclusive entre el acumulador y el dato HHH |
| *CLA* | Pone en 0 el acumulador (limpiar) |
| *CMA* | Invierte los bits del acumulador |
| *INC* | Permite sumar una unidad al dato cargado en el acumulador |
| *JMP HHH* | Salta de instrucción a la palabra HHH |
| *SNA HHH* | Realiza un salto a la palabra HHH si el dato almacenado en el acumulador es negativo |
| *SZA HHH* | Realiza un salto a la palabra HHH si hay un 0 en el acumulador |
| *SCA HHH* | Realiza un salto a la palabra HHH si hay acarreo 1 |
| *INP* | Permite la entrada de un dato desde un periférico de entrada |
| *OUT* | Permite la salida de un dato almacenado en el acumulador hacia un periférico de salida |

1. No, todas por ejemplo la instrucción HLT indica parar el programa y no necesita la lectura ni escritura de ningún dato.
2. Las instrucciones que afectan directamente al IP son:

* SZA
* SCA
* SNA
* JMP

1. Las instrucciones que activan la ALU o status register son:

* ADA
* ANA
* XOA
* INC
* SNA
* SZA
* SCA

# Reloj

1. Un ciclo de clock es el tiempo que transcurre entre dos pulsos adyacentes.
2. Una secuencia repetitiva puede estar formada por 4, 8 o 16 señales de tiempo. El tiempo de una secuencia repetitiva constituye el ciclo de la computadora o ciclo de máquina
3. 1 gHz = 1/1.000.000.000 hertz = cada ciclo de reloj tendrá un tiempo de 10^-9 segundos.
4. 8 ciclos de clock conforman 1 ciclo de máquina, por lo tanto cada ⅛ seg hay un ciclo de clock.