Unidad 3 - Arquitectura del Conjunto de Instrucciones

⊛ Arquitectura de Computadoras

- "Son las características computacionales visibles al programador, es decir, los atributos que tienen impacto directo en la ejecución lógica de un programa"

Ejemplo: Existencia de una instrucción de maquina que permite multiplicar.

- Tener muchas instrucciones podrían perjudicar al programa
- Es conveniente tener procesadores que tengan menos instrucciones es Beneficioso, se hace eficiente el procesador

Habia 2 conjuntos de Computadoras que se nombraron eran las RisK y las Complex

- Agregar Instrucciones complejisa el programa.
  - ISA (Instruction Set Architecture) / Arquitectura de Programación
    - Repertorio de instrucciones
    - Registros
    - Tipos de datos
    - Modos de Direccionamiento
    - Formato de Instrucciones
    - Memoria
      - Word Size
      - B.g / Little Endian
      - Direccionamiento
      - Espacio de Direccionamiento (adress space)

Concepto de Arquitectura: Determina que conjunto de Herramientas concretas para determinar la Logica el programador.

- Repertorio de Instrucciones → Que instrucciones puede realizar mi procesador, o sea el conjunto de instrucciones estaba disponible

- Especificacion de su Operacion → En cada instruccion dicen la instruccion ac sumar, que va a sumar algo aqui esto en "donde" (memoria, registro), especifica que es lo que dice la instruccion para poder ejecutar, o sea, donde estan los operandos para poder cumplir con la instruccion determinado, cuando uno diseña y define el computador.

- Registros: → Registros de Proposito General
  → cantidad de Bits
  → Registro para almacenar una posicion de Memoria

- Tipos de Datos: Poder almacenar en los Registros el tipo de dato, numeros Positivos, Negativos, numeros Naturales.

- Modos direccionamiento: Tiene que ver con una instruccion como va a acceder a su dato, estara en la misma direccion? ¿el dato estara en una instruccion de Memoria? ¿Estara en la misma Instruccion? ¿Estara en otra direccion distinta?
  Son distintos modos en los cuales el RI, mi registro de instruccion va a estar definido,
  Conclucion: hay distintos modos de direccionamiento.

- Formatos de Instrucciones: Representa la manera con que un determinado tipo de dato estara implementado dentro de un Registro o dentro de un lugar de Memoria donde se trabaje, me permitira trabajar con numeros con/sin signo Formato binario, exceso, complemento a la Base.
  Se define que trabajaremos con enteros, Naturales o Reales.
  ¿Como sabemos que son Enteros, Negativos, Positivos o Reales?
  Formato de la instruccion o Formato de 16 bits.
  Hay varios Formatos....

- Memoria : En cada posicion puede ingresar 8 bits

Se ocupo 2 posiciones de Memoria

- En la posición más chica agregar los 8 bits más chicos y en la siguiente los 8 bits más grandes, 8 bits más grandes en la posición más chica y los 8 bits más pequeños en la posicion más grande → Se lo conoce como Big / Little Endian.

Big Endian	o	Little Endian
La direccion más baja me permite poner el dato más grande		La direccion más grande me permite poner el dato más pequeño.
Direccion Alta		Direccion Baja

Ejemplo de Dibujo → Prope

- Espacio de Direcciones : si tenian 8 bits se guardaba en 256 bytes, pero las instrucciones se guardan de a 16 bits o 2 bytes, en 256 bytes puedo acceder a un solo byte, El espacio de direccionamiento me permite dar saltos Instruccion de 32 bits es 4 bytes si estoy en la posicion 00 salto hasta 04 por los 4 bytes o 32 bits.

Todo esto permite favorecer la programación

Ahora Hablamos sobre la Organizacion de Computadoras

- "Implementacion de la arquitectura (microArquitectura). Define los unidades Operativas y sus interconexiones (señales de control, interfaces entre el CPU y los periféricos, tecnologia de Memoria, trayecto del dato, etc).

Ejemplo: Como la instrucción de Multiplicar se ejecuta internamente (por sumas sucesivas u otro circuito electronico)

La Organización es un complemento de esa arquitectura, tiene que ver más con el Hardware, el tipo de Memoria, y los tipos de datos que se van a transferir.

Interface entre CPU y Periféricos, si la CPU tiene "n bits" los periféricos también.

Arquitectura está dirigida a la parte Lógica que tiene relación con el programador

Organización de la Computadora está orientada a "Como eso se va a implementar con distintos tipos de tecnologías

min 39:50 Close Martes 17/10

- Diferentes implementaciones de una misma Arquitectura.
  - Costos
  - Velocidad de Procesamiento
  - Consumo de Energía
- Ejemplo: Intel x86 - Arquitectura Intel 64
  - sistemas Embebidos (smartphone, tablets, gadgets, etc) (Intel Atom)
    - ↳ Bajo Costo y Consumo de Energía
  - Servidores (Intel Xenón)
    - ↳ Alta Velocidad de procesamiento

El modelo de programación de procesadores intel x86

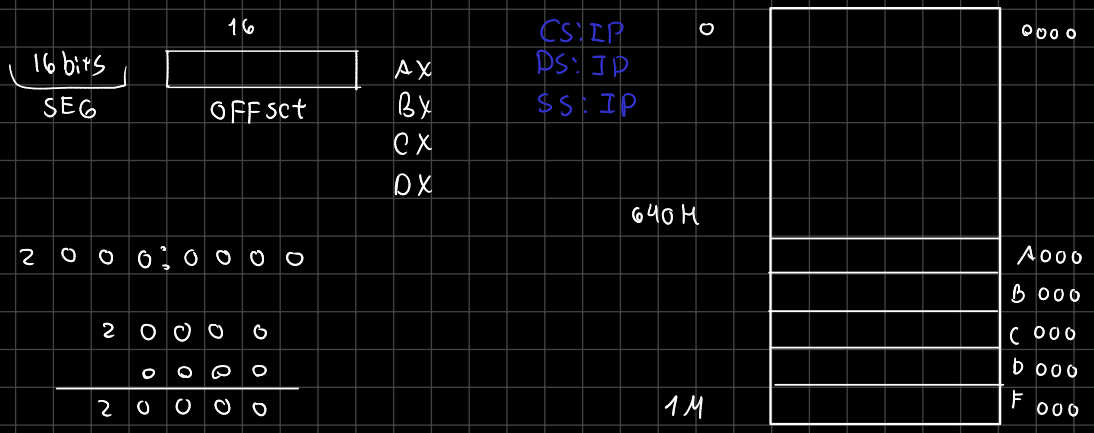
A partir de 2H:28min:00 (clase Martes 17/10)

- Lenguaje de Programación de alto Nivel — C, C++, JAVA, Python
- Lenguaje de Programación de Bajo Nivel. — Lenguaje Ensamblador
- Lenguaje de Programación Maquina. — Lenguaje Maquina

Registros Multiproposito AX, BX, CX, DX

Registros de Proposito Especifico

A partir de 2H:45:00 hasta 3H:10 min: 30seg explica sobre la pizarra, Memoria y un poco de ensamblador (Clase Martes 17/10).



2000:0000

2 0 0 0 0
0 0 0 0
2 0 0 0 0

1800:2000

1 8 0 0 0
2 0 0 0
2 0 0 0 0

En Resumen: 2000:0000 y 1800:2000 son la misma direccion.

segmento 0 — 0000:0000 — 00001 0010

0 0 0 0 0
0 0 0 0
0 0 0 0

segmento 1 — 0001:0000

0 0 0 1 0
0 0 0 0
0 0 0 1 0

En Resumen tengo lo siguiente

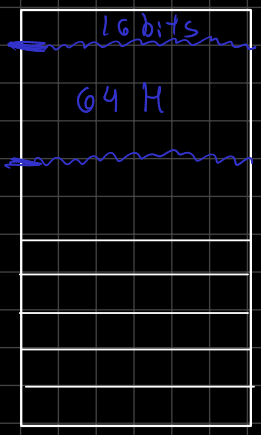
0 0 0 0 1 0 0 1 0 y 0 0 0 1 1 0 0 0 0

0 0 0 0 0
0 0 1 0
0 0 0 1 0 0:2

y

0 0 0 1 0
0 0 0 0
0 0 0 1 0 0:2

En resumen el segmento más pequeño que puedo tener es de 16 bits



0 0 0 0 : 0 0 0 0 0 0 0 0 0 0 0 0 0 16 bits

0 0 0 1 : 0 0 0 0 0 0 0 1 0

64K

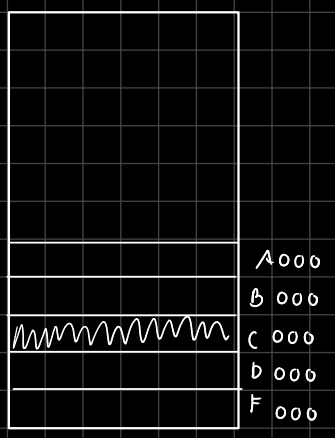
: F F F F

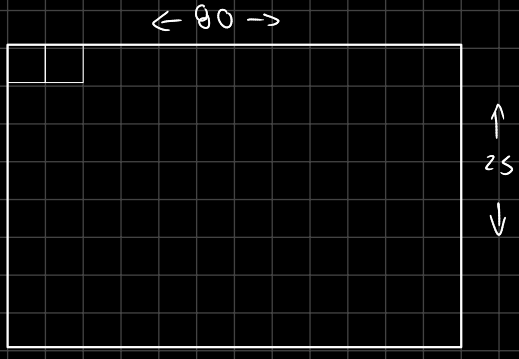
0 0 0 1 0
F F F F
1 0 0 0 F

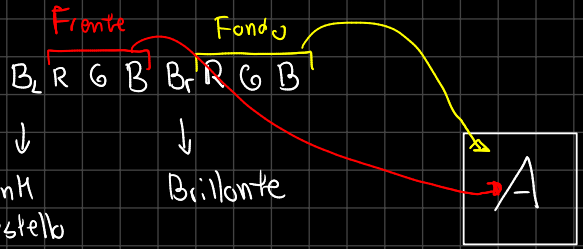
1 0 0 0 : 0 0 0 0

Conclusión: El segmento lo unico que me indica donde hay un grupo de mis datos CS, DS, SS.

B 8 0 0 : 0 0 0 0
B 8 0 0 : 0 0 0 1 -> Define el Color de Fondo
B 8 0 0 : 0 0 0 2







0 1 0 0 0 0 0 1 -> 41 "Define el Frente Rojo y Fondo Azul"

En Resumen tengo | 41 | 41 | Esto me define la letra A = "41h" y el fondo Azul = "41h"

Atributo

En conclusión tenemos lo siguiente "Caracter", "Atributo", "Caracter", "Atributo" ....

CACACA....

A partir 3H:11:00 hablo sobre DosBox (Clase Martes 17/10)

↳ 3H:23:00 reejecuta el programa

A B C D E . . . . . Z

26 Letras con

```
mov   AH, AS
mov   AL, 41
mov   [BX], AX
INC   BX
INC   BX
INC   AL
CMP   BX, 0034
JL    10B
INT   20
```

26 x 2 = 52

48

4

3

16

3

48

0034

AH AL

B_2 R G B B_1 R G B

0 0 0 1 1 0 0 1

1 9

SUB AH, 1

0 1 1 1

7 9

0 1 1 0 1 1 1 1

6 F

Ejercicio de Unidad - 3

1. Escribir la letra a en la memoria de video.

```
Segmento: 0100    MOV   AX, B800
                  MOV   DS, AX
                  MOV   BX, 0000
                  MOV   AX, 4141
                  MOV   [BX], AX
                  INT   20
```

2. Llenar una fila de la pantalla de la consola con la letra a.

```
Seg: 0100    MOV   AX, B800
             MOV   DS, AX
             MOV   AX, 4141
             MOV   [BX], AX
             INC   BX
             INC   BX
             CMP   BX, 00A0
             INT   20
```

3. Escribir una fila con el abecedario.

```
Seg: 0100          MOV   AX, B800
                   MOV   DS, AX
                   MOV   BX, 0000
                   MOV   AX, 4141
        010B       MOV   [BX], AX
                   INC   BX
                   INC   BX
                   CMP   BX, 0034
                   JL    ,10B
                   INT   20
```

4. Escribir todo el abecedario con un atributo diferente para cada letra.

```
Segmento: 0100    MOV   AX, B800
                  MOV   DS, AX
                  MOV   BX, 0000
                  MOV   AX, 6F41
          010B    MOV   [BX], AX
                  INC   BX
                  INC   BX
                  INC   AL
                  SUB   AH, 1
                  CMP   BX, 0034
                  JL    10B
                  INT   20
```

5. Pintar la pantalla de la consola con los colores de tu equipo de futbol favorito.

```
Seg: 0100    MOV   AX, B800
             MOV   DS, AX
             MOV   BX, 0000
             MOV   AX, 7741
Seg: 010B    MOV   [BX], AX
             INC   BX
             INC   BX
             CMP   BX, 07D0
             JL    010B
             MOV   BX, 07D1
             MOV   AX, 4044
Seg: 011B    MOV   [BX], AX
             INC   BX
             INC   BX
             CMP   BX, 0FA0
             JL    011B
             INT   20
```

6. Escribir “Organización del Computador” sin acentos en la memoria de video con letra roja sobre fondo amarillo.

Vamos a programar

```
ORG   100h

      MOV AX, 3
      INT 10h

      MOV AX, 1003h
      MOV BX, 0
      INT 10h

      MOV AX, 0B800h
      MOV DS, AX

; A partir de Aqui agrasamos el texto caracter a caracter de "Organizacion del Computador"

      MOV [02h], 'O'
      MOV [04h], 'r'
      MOV [06h], 'g'
      MOV [08h], 'a'
      MOV [0Ah], 'n'
      MOV [0Ch], 'i'
      MOV [0Eh], 'z'
      MOV [10h], 'a'
      MOV [12h], 'c'
      MOV [14h], 'i'
      MOV [16h], 'o'
      MOV [18h], 'n'
      MOV [1Ah], ' '
      MOV [1Ch], 'd'
      MOV [1Eh], 'e'
      MOV [20h], 'l'
      MOV [22h], ' '
      MOV [24h], 'C'
```

5 Nuevas lineas de codigo que permiten el correcto uso de mi programa.

Codigo habitual para hacer que Funcione correctamente

```
MOV [26 h], 'o'
MOV [28 h], 'm'
MOV [2A h], 'p'
MOV [2C h], 'u'
MOV [2E h], 't'
MOV [30 h], 'a'
MOV [32 h], 'd'
MOV [34 h], 'o'
MOV [36 h], 'r'
; A partir de Aqui vamos a analizar sobre el Registro CX y DI
; CX almacenara el dato de la cantidad de_ iteraciones que
                tendra las palabras "Organizacion del Computador"
; DI almacena la direccion de inicio de los atributos de los
; caracteres
 MOV CX, 27
 MOV DI, 03h          Fondo  letra
iterador: MOV DI, 1110 1100b ; Almacena el codigo en binario
                               Fondo  Letra
          ADD DI, 2
          LOOP iterador
```

Aprendemos sobre 5 nuevas instrucciones

2 instrucciones para el Contador CX de loop y

DI que almacena la direccion del primer atributo

3 Instrucciones Agregar el Color

1 para el Color en Atributos
1 para el Bumodo 20)
1 para el iterador → loop

```
MOV AH, 0   } // Espera que se apriete una tecla para pasarle
INT 16h     }    el control al sistema operativo.
RET
```

Ejercicio Escribir "Hola Mundo!" de Fondo Amarillo y Letra Roja

```
ORG 100h

    MOV AX,3
    INT 10h

    MOV AX,1003h
    MOV BX,0
    INT 10h

    MOV AX,0B800h
    MOV DS,AX

    MOV [02h],"H"
    MOV [04h],"o"
    MOV [06h],"l"
    MOV [08h],"a"
    MOV [0Ah]," "
    MOV [0Ch],"M"
    MOV [0Eh],"u"
    MOV [10h],"n"
    MOV [12h],"d"
    MOV [14h],"o"
    MOV [16],"!"

;A partir de Aqui se analiza sobre la carga de Iteracion y la direccion
;en DI que es 03h el primero de los Atributos
    MOV CX,11
    MOV DI,03h

;A partir de Aqui se trata de acceder a los atributos y asignarle
                                        color a cada uno!
iterador: MOV [DI], 1110 1100b
          ADD DI,2
          LOOP iterador
```

1 2 3 4 5 6 7 24

Ejercicio "Adios Mundo y Hola Mundo" Fondo y Letra
1111 0100

```
ORG 100h

    MOV AX,3
    INT 10h

    MOV AX,1003h
    MOV BX,0
    INT 10h

    MOV AX,0B800h
    MOV DS,AX

    MOV [02h],"A"
      :
    MOV [30h],"o"
```

```
        MOV   CX, 24
        MOV   DI, 03h

iterador: MOV  [DI], 11110100b
        ADD   DI, 2
        LOOP  iterador

        MOV   AH, 0
        INT   16h

RET
```

7. Escribir en la pantalla la palabra microprocesador y a continuación la misma palabra invertida de derecha a izquierda.

El programa es Simple

```
; Analizamos el Bloque de directivas

.model small

.stack 64

.DATA
        cad1  db  "Cadena de Prueba", "$"
        cad2  db  16 dup(" "), "$"      ; Reserva 16 bits de cadena

.CODE

inicio:  ; Punto de entrada del Programa

        MOV   AX, @data
        MOV   DS, AX
        MOV   ES, AX

        MOV   CX, 16    ; Longitud de la Cadena

        LEA   SI, cad1  ; Mueve a SI el OFFSET de cad 1

        LEA   DI, cad2 + 15  ; Apunta al final del área reservada
                             para almacenar la cadena invertida
otro:
        LODSB ; Obtiene el caracter de cad1, guarda en AL y actualiza

        MOV   [DI], AL  ; Almacena el caracter leído de cad1
                        ; en la posición actual de DI. Posición final
                        ; de cad2 para el primer caracter

        DEC   DI        ; Decrementa a DI en 1

        LOOP  otro      ; Comienza hasta que termine el Iterador


        LEA   DX, cad1

        MOV   AH, 09h   ; Imprimir cadena original
```

Se carga una dirección de Memoria en REG -> SI y DI

```
        INT   21h
        LEA   DX, cad2   ; Mueve a dx el offset de cad 2
        MOV   AH, 09h    ; Imprimir Cadena Invertida

        MOV   AX, 4C00h  ; Terminar Programa y Regresar al DOS
        INT   21H
END     inicio

END
```

```
. model small
. stack 64
. DATA
      cad1  db  "Cadena de Prueba", "$"
      cad2  db  16 dup(" "), "$"      ; Pesos
. CODE
inicio:


END inicio
END
```

1°)

Las Directivas, la cad1 y la reserva de cad 2

```
. model small
. stack 64
. DATA
      cad1 db "Cadena de Prueba", "$"
      cad2 db  16 dup(" "), "$"
. CODE
inicio:
      ⋮
END inicio
END
```

```
. model small
. stack 64
. DATA
      cadena_1  db  "Alexander Cruz", "$"
      cadena_2  db  14 dup(" "), "$"
. CODE
inicio


END inicio
END
```

1°) Estructura

2°) Imprimir Registros

3°) Longitud de Cadena y guardarlo en cadena.

3° Guardado de Cadena

9. Un byte se encuentra en la dirección 0100. Ubicar en la dirección 0101 su nible más significativo.

```
ORG  100h
    MOV  AL, 10110010b
    MOV [100], AL

    MOV  AL, [0100]
    AND  AL, 11110000b
    MOV [101], AL
RET
```

Por si se quiere cargar un dato en la dirección de memoria [100]

10. En las direcciones 0100 y 0101 hay dos enteros positivos. Ubicar en la dirección 0102 el menor de los dos.

```
ORG  100h

    MOV  AX, 00FFh
    MOV  [100], AX
    MOV  AX, 0001h
    MOV  [101], AX

    MOV  AX, [100]
    MOV  BX, [101]
    CMP  AX, BX
    JLE  A_Mayor_B
    JB   B_Mayor_A

A_Mayor_B:
    MOV  [102], BX
    JMP  FIN

B_Mayor_A:
    MOV  [102], AX
    JMP  FIN

FIN:
RET
```

11. A partir de la dirección 0200 hay un número de 3 bytes de longitud. Luego de ese número, en la dirección siguiente de la memoria, hay otro número de 24 bits. Sumar los dos números y guardar el resultado a partir de la dirección posterior al segundo número.

14. Sumar el conjunto de datos de 16 bits que comienzan en la dirección 203 y tiene una longitud dada por el dato almacenado en la dirección 202. Elegir los datos para que su suma sea menor a 65.536. Almacenar el resultado a partir de la dirección 200.

```
ORG   100h

      MOV   AX, 3
      MOV   [202], AX

      MOV   AX, 1000 1101 0000 1111b
      MOV   [203], AH
      MOV   [204], AL
      MOV   AX, 0001 1100 1010 0101b
      MOV   [205], AH
      MOV   [206], AL
      MOV   AX, 0000 0000 0000 1111b
      MOV   [207], AH
      MOV   [208], AL
      MOV   AX, 0
      MOV   BX, 0
      MOV   CX, 0
```

15. A partir de la dirección 204 hay un conjunto de datos de longitud igual al dato almacenado en la dirección 203. Almacenar en la dirección 200 la cantidad de datos que tienen el bit 3 en 1.

```
ORG  100 h

     set:
        MOV   AX, 3
        MOV  [203], AX
        MOV   AX, 1000 1101 0000 1111b
        MOV  [204], AH
        MOV  [205], AL
        MOV   AX, 0001 1100 1010 0101b
        MOV  [206], AH
        MOV  [207], AL
        MOV  AX, 0000 0000 0000 1111b
        MOV  AX, 0
        MOV  BX, 0
        MOV  CX, 0
        JMP  inicio
        inicio:
```

16. A partir de la dirección 204 hay un conjunto de datos de longitud igual al dato almacenado en la dirección 203.
Almacenar en la dirección 200 la cantidad de datos igual a cero, en la 201 la cantidad de datos positivos y en la 202 la cantidad de datos negativos.

```
ORG   100h

    set:
      MOV  AX, 3
      MOV [203], AX
      MOV  AX, 1010 1110 0101 1111b
      MOV [204], AH
      MOV [205], AL
      MOV  AX, 1000 1111 0000 0111b
      MOV [206], AH
      MOV [207], AL
      MOV  AX, 0011 0101 1110 1111b
      MOV [208], AH
      MOV [209], AH
      MOV  AX, 0
      MOV  BX, 0
      MOV  CX, 0
      JMP inicio

        inicio:
```

17. A partir de la dirección 3002 hay un conjunto de datos de longitud dada en la dirección 3001. Almacenar en la dirección 3000 el dato menor.

```
ORG  100h

    ; Carga de Datos

    set:
        MOV  AX, 3
        MOV  [3001], AX

        MOV  AX, 11110101b
        MOV  [3002], AX

        MOV  AX, 11010101b
        MOV  [3003], AX

        MOV  AX, 11010111b
        MOV  [3003], AX

    ; Inicialización

        MOV  CX, 0        ; inicializa CX en 0
        MOV  AL, [3001]   ; AL guarda la "cantidad" de datos guardada en [3001]
        MOV  CL, AL       ; CL tendra el contador en la cantidad almacenada [3001]

        MOV  SI, 3002h    ; El registro SI apunta a la dirección del 1er Dato.
        MOV  AL, [SI]     ; Primer Dato almacenado en AL
        MOV  BL, AL       ; BL tiene el primer Dato minimo para comparar

    ; Loop de Comparación

    buscar_menor:

        MOV  AL, [SI]
        CMP  AL, BL       ; Compara si AL >= BL si se cumple Salta (caso contrario continua)
        JAE  siguiente

        MOV  BL, AL       ; Si BL es menor que AL, por lo tanto se actualiza BL

    siguiente:

        INC  SI
        DEC  CL
        JNC  buscar_menor ; Si CL ≠ 0 entonces salta por la condición

    ; Guarda el Resultado

        MOV  [3000], BL   ; Guarda el resultado en

        RET
```

4Linea → buscar_Menor Siguiente = 3

Seteo de Datos → Inicialización → Loop de Comparación → Guardado

8 lineas 6 Linea 7 linea importantes 1 Linea

18. Almacenar en la dirección 200 la cantidad de unos que tiene el dato almacenado en la dirección 201.

```
ORG   100 h
set:
      MOV   AX, 1101 0000 1111 1010 b
      MOV   [201], AX

Programa:

      MOV   C2, 16
      MOV   D1, 0
```

Unidad 3 - Memoria de VIDEO VGA

Memoria de Video VGA

Monitor de Pantalla de Cristal Liquido (LCD) digital directo recibe un flujo de bits digitales directamente desde el controlador de un video, y no requiere del barrido de trama. Por lo general, las pantallas digitales muestran un texto más fino que las pantallas Analogicos

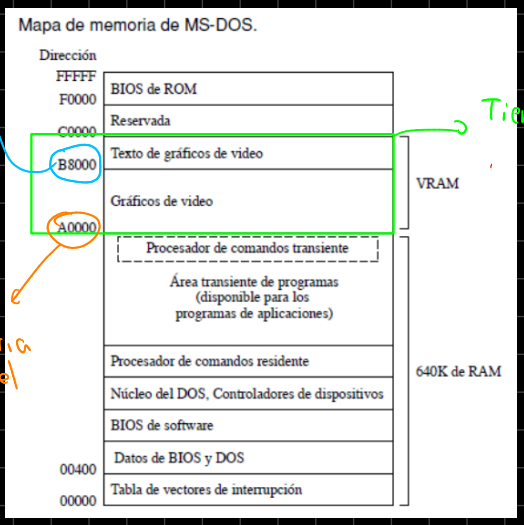
La VRAM almacena datos de VIDEO. La VRAM tiene doble puerto, y permite que un puerto actualice en forma Continua la pantalla, mientras que el otro puerto escribe datos.

¿Qué es la Memoria de VIDEO?

Es el área de la memoria de VIDEO (VRAM) en una "IBM-PC" empieza en la ubicación A0000, la cual se utiliza cuando el adaptador de video cambia al modo de gráficos. Cuando el video se encuentra en modo de texto a color, la ubicación de Memoria B8000 almacena todo el texto que se muestre en la pantalla. La pantalla representa un mapa en la memoria, de manera que cada fila y columna en la pantalla corresponde a una palabra de 16 bits en la memoria

Cuando se copia un caracter a la memoria de Video, aparece de inmediato en la pantalla.

Mapa de Memoria de MS-DOS



Es la ubicación de Memoria B8000 almacena todo el texto que se muestra en pantalla.

Tiene Relación con la VRAM que es parte de la GPU.

Ubicación de Memoria cuando se utilizan el adaptador de VIDEO a modo Gráficos.

Una interrupción común es INT 10H (Servicios de VIDEO). Son procedimientos que muestran rutinas que controlan la posicion del cursor, escriben el texto a color, desplazan la pantalla y muestran graficos de VIDEO

Programación de VIDEO con INT 10h (Se habla sobre la programación)

Acceso a nivel de MS-DOS: Cualquier computadora que ejecute o emule a MS-DOS puede utilizar la INT 21h para escribir texto en la pantalla de VIDEO.

Acceso Directo al VIDEO: Los caracteres se mueven directamente a la RAM de VIDEO, por lo que la ejecución es Instantanea. Durante la era MS-DOS, los programas procesadores de palabras y las hojas electronicas de Calculo utilizaban este método.

Funcionamiento del texto de VIDEO