Diretivas
DB (define byte)
DW (define word)
DUP (duplicate)
EQU (constante)
ORG (seta o endereço de memória onde vai ter início aquele segmento)
SEGMENT (inicia um segmento)
ENDS (finaliza um segmento)
ASSUME (faz a ligação entre os segmentos e os registradores de segmento)
END (indica o label onde tem início a execução do programa)
.MODEL
.DATA
.CODE
OFFSET (endereço efetivo, deslocamento)
BYTE PTR
WORD PTR
SEG (endereço do segmento)

Instruções

Transferência de dados
MOV *Destino, Fonte*
MOV *BYTE PTR [Reg. Indice], Fonte*
MOVSX *Destino, Fonte* (sinal estendido)
MOVZX *Destino, Fonte* (zero estendido)
PUSH *Fonte* (coloca palavra na pilha, fonte=reg, SP = SP-2)
PUSHW *Fonte* (fonte = imediato)
PUSHD *Fonte* (dword)
PUSHA (AX, CX, DX, BX, SP, BP, SI, DI)
POP *Destino*
POPA
IN *Acumulador, Porta* (lê byte ou palavra da porta)
INS *BYTE PTR ES:[DI], DX* (dest = pos. memória)
INSB
OUT *Porta, Acumulador* (envia byte ou palavra para a porta)
OUTS *DX, BYTE PTR DS:[SI]*
OUTSB
LEA *Destino, Fonte* (carrega end. efetivo)
PUSHF (coloca o reg. de flags na pilha)
POPF
XCHG *Destino, Fonte*
BSWAP *Destino* (reg. 32 bits: 1=4; 2=3; 3=2; 4=1)
XLAT (AL = DS:[BX+AL])
LDS *Destino, Fonte* (dest=DS:[fonte]; DS=DS:[SI+2])
LAHF (AH = flags)

Strings
CLD / STD (limpa/carrega flag direção)
REP / REPE / REPZ / REPNE / REPNZ (enquanto CX > 0)
MOVS *String-destino, String-fonte* (move string)
MOVSB (byte por byte) / MOVSW
CMPS *String-destino, String-fonte* (compara string)
CMPSB / CMPSW

Teoria:

O conjunto de ações executadas quando ocorre uma solicitação de interrupção mascarável (de hardware).

Quando um dispositivo necessita efetuar alguma operação (geralmente de entrada/saída) ele envia um sinal (IRQ), em seguida o 8259A põe sua saída (INT) em nível alto. Esta saída é conectada ao microprocessador no pino INTR, este pino é usado pelo microprocessador para sinalizar uma interrupção mascarável. Se o bit de interrupção (IF) do registrador FLAGS estiver setado, o microprocessador envia um sinal (INTA) de volta ao 8259A. Ao receber este sinal, o controlador coloca um número inteiro no barramento, este número é usado para identificar o tipo de dispositivo e é chamado de vetor de interrupção. A CPU então usa este número para indexar uma tabela de 256 entradas para encontrar o endereço da rotina de tratamento de interrupção.

Conjunto de ações interrupção por software:

- Salva Estado
- Processa Interrupção
- Restaura Estado Retorno
- POP IP, POP FLAGS

Descreva os tipos de interrupções do processador:

Por Software:

- Uma interrupção de software é apenas uma instrução especial de um programa que esteja controlando o microprocessador. Em vez de somar, subtrair ou coisa que o valha, a interrupção de software faz com que a execução do programa seja desviada temporariamente para outra seção de código na memória.
- Intruções INT
- Sistema Operacional.
- Usuário.

Por Hardware:

- É controlada por sinais especiais externos ao fluxo de dados normal. O único problema esta em que os microprocessadores reconhecem muito menos interrupções do que seria desejável são apenas duas as linhas de sinais de interrupção. Uma delas é um caso especial: a NMI (interrupção não mascarável). A outra é compartilhada por todas as interrupções do sistema.
- NMI, IRQ´s conectadas ao pino INTR.

SCAS *String-destino* (compara com AL,AX,EAX)
SCASB / SCASW
LODS *String-fonte* (carrega string em AL,AX,EAX e atualiza SI para o próximo elemento)
STOS *String-destino* (escreve o valor de AL,AX,EAX em ES:DI e atualiza DI)

Aritméticas
ADD *Destino, Fonte*
ADC *Destino, Fonte* (com carry)
INC *Destino*
SUB *Destino, Fonte*
SBB *Destino, Fonte* (com borrow)
DEC *Destino*
CMP *Destino, Fonte*
CMPXCHG *Destino, Fonte* (se dest==AL,AX,EAX, entao dest=fonte, senão AL,AX,EAX=dest)
MUL *Fonte* (sem sinal, AX=fonte x AL ou DX:AX = fonte x AX)
IMUL *Fonte* (com sinal)
DIV *Fonte* (sem sinal, AX/fonte, AL=quociente, AH=resto)
IDIV *Fonte* (com sinal)
NEG *Destino* (0 - dest)
CBW (converte o AL em AX, com sinal estendido)
CWD (AX em DX:AX)
DAA (ajuste decimal para adição, BCD, muda o valor de AL)
DAS (subtração)
AAA (ajuste ASCII para adição)
AAS (subtração)
AAM (multiplicação)
AAD (divisão)
XADD *Destino, Fonte* (fonte=dest; dest=dest+fonte)

Lógicas
NOT *Destino*
AND *Destino, Fonte*
OR *Destino, Fonte*
XOR *Destino, Fonte*
TEST *Destino, Fonte*
SETA / SETAE / SETB / SETBE / SETC / SETE / SETG / SETL / SETO / SETP / SETS / SETZ
(a= acima, e = igual, b = abaixo, c = carry, g = maior, l = menor, o = overflow, p = paridade, s = sinal, z = zero)

Manipulação de Bit
SHL Destino, Quantidade (shift left lógico, carry = bit que sai)
SAL Destino, Quantidade (shift left aritmético)
SHR Destino, Quantidade (shift right lógico)
SAR Destino, Quantidade (shift right aritmético)
SHLD / SHRD Destino, Fonte, Quantidade (dupla precisão, ao invés de zeros, coloca os bits da fonte para o destino)
ROL / ROR Destino, Quantidade (carry = bit que rotaciona)
RCL / RCR Destino, Quantidade (bit do carry entra e carry=bit que sai)

Laços e Desvios
JMP Target
JMP FAR PTR Local (intersegmento)
JMP SHORT Local (-128 a 127 bytes)

- Internas.
- Externas (Mascarável, não- mascarável).
- As interrupções não-mascaráveis são usadas para sinalizar “quase catástrofes”, como um erro de paridade de memória. Todos os dispositivos de E/S utilizam interrupções mascaráveis.

A funcionalidade das unidades que compõem os pipelines Pentium (básico) e a função de cada registradores de uso geral e de segmento:

Pipeline: O processador é dividido em 2 pipelines de inteiro: U e V. Eles são responsáveis por executar as instruções do processador.

Funcionalidade Unidades:

PF (Prefetch): A CPU busca o código da cachê de instruções e alinha o código para o byte inicial da próxima instrução a ser codificada.

D1 (Primeira Decodificação): A CPU decodifica a instrução para gerar uma palavra de controle. Uma única palavra de controle executa instruções diretamente.

D2 (Segunda Decodificação): A CPU decodifica a palavra de controle de D1 para usar no próximo estágio. A CPU gera endereços p/ referência de dados na memória.

EX (Execução): A CPU pode acessar a cachê de dados ou calcula resultados na ULA.

Escrita (Write Back): A CPU atualiza os registradores e flags com o resultado das instruções.

Superescalar: Habilita 2 instruções a serem executadas em paralelo. Os recursos para geração de endereço foram replicados em pipelines independentes chamados de U e V.

Registradores de Segmentos:

São 5 registradores: CS, DS, SS, ES, FS E GS. (16 bits) controlado pelo programador. Servem para sofrerem um shift left e serem somados com outro registrador e gerar o endereço a ser usado.

JA --> (CF ou ZF) = 0
JAE --> CF = 0
JB --> CF = 1
JBE --> (CF ou ZF) = 1
JBNE --> (CF ou ZF) = 0
JC --> CF = 1
JE --> ZF = 1
JG --> ((SF xor OF) ou ZF) = 0
JGE --> (SF xor OF) = 0
JL --> (SF xor OF) = 1
JLE --> ((SF xor OF) ou ZF) = 1
JNA --> (CF ou ZF) = 1
JNAE --> CF = 1
JNB --> CF = 0
JNC --> CF = 0
JNE --> ZF = 0
JNG --> ((SF xor OF) ou ZF) = 1
JNGE --> (SF xor OF) = 1
JNL --> (SF xor OF) = 0
JNLE --> ((SF xor OF) ou ZF) = 0
JNO --> OF = 0
JNP --> PF = 0
JNS --> SF = 0
JNZ --> ZF = 0
JO --> OF = 1
JP --> PF = 1
JPE --> PF = 1
JPO --> PF = 0
JS --> SF = 1
JZ --> ZF = 1
LOOP Label (enquanto CX > 0)
LOOPE / LOOPZ / LOOPNE / LOOPNZ
JCXZ (jump se CX = 0)
CALL Procedimento (PUSH IP e PUSH CS(intersegmento))
RET (retorno de procedimento, POP IP e POP CS)
INT Tipo de Interrupção (PUSH IP, PUSH CS e PUSHF)
IRET (retorno de interrupção)
INTO (se OF = 1)

<u>Controle</u>
CLC (CF = 0)
STC (CF = 1)
CMC (CF = complemento de CF)
CLD (DF = 0)
STD (DF = 1)
CLI (IF = 0)
STI (IF = 1)
HLT (parar até interrupção ou reset)
NOP
LOCK (bloqueia barramento durante a próxima instrução)

Registradores de Uso Geral:
São 7 registradores: AX, BX, CX, DX, BP, SI e DI.
AX(acumulador): usado para multiplicações, divisões e instruções que acessam portas.
CX(contador): usado para operações de loop e CL (8b) é usado para rotações e shift.
DX(dados): usado em operações de multiplicação e divisão e como ponteiros para acessos de I/O.
SI (source index) e DI (destination index): usados em manipulação de strings(ponteiros).

Assembly:

```
criandoArquivo:
                  mov ah, 3ch
                  xor cx,cx
                  lea dx, nomeArquivo
                  mov cx, 01h
                  mov manipulador, ax
                  int 21h

Array:
               vetor db 10 dup(?)


Manipulando Array e lendo do teclado:
               xor si, si
               mov [vetor + si], al
               inc si


fechaArquivo:
               mov ah, 3Eh
               int 21h


FinalizaPrograma:
                        mov ax, 4c00h
                        int 21h

IniciaPrograma:
                        mov ax, @data
                        mov ds, ax
```

Interrupções
As funções são colocadas em AH
Teclado
DOS INT 21H, Função 01H (Wait for keyboard input: espera pressionar tecla e mostra na tela, AL recebe o ASCII. Se AL = 0, tem que chamar INT 21H de novo pra pegar o ASCII estendido)
DOS INT 21H, Função 08H (Console input without echo: espera pressionar tecla, mas não mostra na tela)
BIOS INT 16H, Função 00H (Read keyboard input: espera pressionar tecla, ASCII em AL, se AL=0, ASCII estendido em AH, não mostra na tela)
BIOS INT 16H, Função 01H (Read keyboard status: espera qualquer tecla, ASCII em AL, se AL=0, ASCII estendido em AH, ZF =1 se não pressionar nada, ZF = 0 caso contrário)
BIOS INT 16H, Função 02H (Return shift flag status: retorna quais teclas estão pressionadas AL[7:insert, 6:caps, 5:num, 4:scroll, 3: alt, 2:ctrl, 1:shift esq, 0:shift dir])

Vídeo
DOS INT 21H, Função 02H (Display output: caractere deve ser colocado em DL)
DOS INT 21H, Função 09H (Display string: DX deve ter o endereço inicial da string)
BIOS INT 10H, Função 00H (Set video mode: AL deve ter o modo de vídeo (0H,1H,...,12H))
BIOS INT 10H, Função 01H (Ajusta formato do cursor: CH=linha inicial, CL=linha final)
BIOS INT 10H, Função 0FH (Read current video mode: retorna modo em AL, BH página ativa)
BIOS INT 10H, Função 02H (Set cursor position: linhas em DH e colunas em DL, BH n° da página)
BIOS INT 10H, Função 03H (Read current cursor position: BL n° da página, retorna DH linhas e DL colunas, CX tipo do cursor)
BIOS INT 10H, Função 05H (Selecionar página: AL=página)
BIOS INT 10H, Função 0AH (Write character to screen: ASCII deve estar em AL, BH o n° da página, CX n° de caracteres)
BIOS INT 10H, Função 0CH (Escreve um ponto (pixel): AL=cor, BH=página, CX=coluna, DX=linha)
BIOS INT 10H, Função 0DH (Lê um ponto: recebe a cor em AL)
BIOS INT 10H, Função 09H (Write character/attribute to screen: AL=ASCII code, BH=n° da página, BL=atributo do caractere, CX=n° de caracteres)
BIOS INT 10H, Função 08H (Read character/attribute from screen:)
BIOS INT 10H, Função 06H (Scroll current page up: AL=n° de linhas (0 p/ tela toda), BH=atributo, CH=n° da linha do topo, CL=n° da coluna no topo esquerdo, DH=n° da linha da base, DL=n° da coluna na base direita)
BIOS INT 10H, Função 07H (Rolar para baixo)

Impressora
DOS INT 21H, Função 05H (Printer output: DL=ASCII do caractere)
BIOS INT 17H, Função 00H (Print character: AL=ASCII do caractere, DX=n°da impressora (0 a 2),retorna status em AH)
BIOS INT 17H, Função 01H (Initialize printer: DX=n° da impressora, retorna status em AH)
BIOS INT 17H, Função 02H (Read printer status: DX=n° da impressora, retorna status em AH)

Warm Start → INT 19H

Timer
BIOS INT 1AH, Função 00H (Lê contador do relógio: retorna AL=0, se não passaram 24h da última leitura, CX:DX valor do contador)
BIOS INT 1AH, Função 01H (Atualiza contador do relógio: CX:DX contador)
BIOS INT 1AH, Função 02H (Lê relógio: retorna CH=horas em BCD, CL=min em BCD, DH=seg em BCD, DL=miliseg em BCD)
BIOS INT 1AH, Função 03H (Atualiza relógio: CH, CL, DH, DL como acima)

Exemplo de cabeçalho:

```
.model small
.stack
.data
        nomeArquivo db "arquivo.txt", "$"
        manipulador dw ?
        limiteVelocidade db 5d
        distancia db 40d
        primeiroTempo dw ?
        segundoTempo dw ?
        diferencaTempo dw ?
        velocidade db ?
        teste db "eric","$0"
      MensagemVelocidade db "Velocidade (m/s): "
      lenVel equ $ - offset MensagemVelocidade

      Hun db ?
      Ten db ?
      One db ?
      quebraLinha DB 13, 10
      len equ $ - offset quebraLinha
```

Teclado

```
armazenar_char: push ebp
                mov ebp,esp

                mov eax,[ebp+8]     ;eax = &ultimo_char
                mov [end_char],eax  ;

                mov eax,101
                mov ebx,300h
                mov ecx,4
                mov edx,1
                int 0x80        ;usando IO_perm para entrada

pegar_ack:
                 mov dx,300h
                in al,dx
                cmp al,ack
                jnz pegar_ack

                in al,dx
                mov ebx,[end_char]
                mov [ebx],al




                mov eax,101
                mov ebx,301h
                mov ecx,4
                mov edx,1
                int 0x80        ;usando IO_perm para saida

                inc dx
                out dx,0         ;zerando o buffer
```

BIOS INT 1AH, Função 04H (Lê data: retorna CH=século em BCD, CL=ano em BCD, DH=mês em BCD, DL=dia em BCD)
BIOS INT 1AH, Função 05H (Atualiza data)

Serial → INT 14H

Disco
Sempre limpa CF se não houver erro
BIOS INT 13H, Função 00H (Reset diskette system: DL=n° do drive (0[A] ou 1[B]), CF=0 entao sucesso, senão AH tem código de erro)

Código	significado
00H	sem erro
01H	função inválida
02H	endereço não encontrado
03H	erro de proteção de escrita
04H	setor não encontrado
06H	linha de mudança do disquete ativa
08H	erro de DMA
09H	erro na data
0CH	tipo de mídia não encontrado
10H	erro no ECC ou CRC
20H	falha no controlador geral
40H	falha na operação de seek
80H	timeout

BIOS INT 13H, Função 01H (Read diskette status: DL=n°do drive)
BIOS INT 13H, Função 02H (Read diskette sector: AL=n° de setores, CH=n° da trilha, CL=n° do setor, DH=n° da cabeça (0 ou 1), DL=n°do drive (0 ou 1), BX=apontador pro buffer, ES=segmento do buffer)
BIOS INT 13H, Função 03H (Write diskette sector:)
BIOS INT 13H, Função 04H (Verifica disquete)
BIOS INT 13H, Função 05H (Formata)
DOS INT 21H, Função 47H (Get current directory: DL=n° do drive (1,2 ou 3), DS:SI tem q apontar pra área de dados, CF=0 sucesso, senão AX tem código de erro)

Error code	meaning
00H	successful
01H	invalid function number
02H	file not found
03H	path not found
04H	no more handles available
05H	access denied
06H	invalid handle
07H	bad memory control blocks
08H	insufficient memory
09H	invalid memory block address
0AH	invalid environment
0BH	invalid format
0CH	invalid access code
…	
1FH	general failure

Continuação teclado

```
                mov eax,101
                mov ebx,301h
                mov ecx,4
                mov edx,0
                int 0x80        ;tirando permissao

                mov eax,101
                mov ebx,300h
                mov ecx,4
                mov edx,0
                int 0x80



                ret
```

DOS INT 21H, Função 3BH (Set current directory: DS:DX tem que apontar pra uma ASCIIZ string, indicando o caminho)
DOS INT 21h, Função 39H (Create subdirectory: DS:DX ¨)
DOS INT 21H, Função 3AH (Delete subdirectory: DS:DX ¨, diretório tem que estar vazio)
DOS INT 21H, Função 19H (Get current drive: retorna o drive em AL)
DOS INT 21H, Função 3DH (Open file with handle: DS:DX ¨, AL=código de acesso)
DOS INT 21H, Função 3FH (Read from file: BX=handle do arquivo, CX=n°de bytes, DS:DX aponta pra DTA, retorna n° de bytes lidos em AX ou código de erro)
DOS INT 21H, Função 3EH (Close file with handle: BX=handle, retorna uma cópia do handle em AX ou código de erro)
DOS INT 21H. Função 3CH (Create file: DS:DX apontando pro ASCIIZ, CX=atributo do arquivo)

Atributo	significado
00H	normal
01H	somente leitura
02H	oculto
03H	sistema

DOS INT 21H, Função 40H (Write to file: BX=handle, CX=n° de bytes pra escrever, DS:DX apontando pra DTA, retorna o n° de bytes escritos em AX senão código de erro)
DOS INT 21H, Função 42H (Position file pointer: BX=handle, AL=código de método, offset em CX:DX, retorna apontador para o arquivo em DX:AX)

Código	significado
0	offset absoluto do byte desde o início do arquivo
1	offset relativo do byte desde a posição atual
2	offset absoluto do byte desde o fim do arquivo

DOS INT 21H, Função 43H (Get or set attributes: AL=0(get) ou 1(set), CX=novo valor do atributo, DS:DX apontando pra um ASCIIZ, retorna atributo antigo em CX, em caso de set)
DOS INT 21H, Função 56H (Rename file: DS:DX aponta para ASCIIZ atual, ES:DI aponta pra nova ASCIIZ)
DOS INT 21H, Função 57H (Get or set file date and time: AL=0(get) ou 1(set), CX=hora, DX=data, retorna hora em CX e data em DX, em caso de get)
DOS INT 21H, Função 1AH (Set disk transfer área (DTA): DS:DX apontado pra nova DTA (80H geralmente))
DOS INT 21H, Função 1BH (Get current drive information: retorna AL=setores por unidade de alocação, CX=bytes por setor, DX=n°de unidade de alocação, DS:BX apontando pra FAT ID byte)
DOS INT 21H, Função 1CH (Get drive information: DL deve possuir o n°do drive, igual à anterior)
DOS INT 21H, Função 4EH (Find file: DS:DX deve apontar pro ASCIIZ da string do arquivo que vai ser procurado, retorna AX=0, se sucesso)
DOS INT 21H, Função 41H (Delete File: DS:DX ¨)