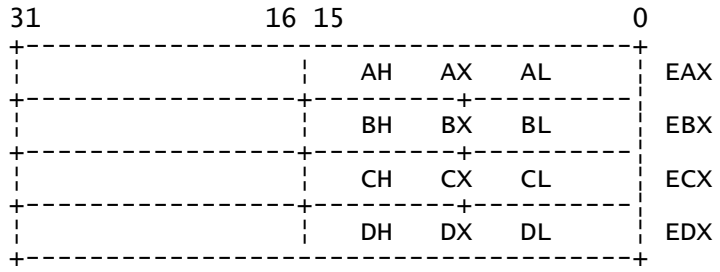
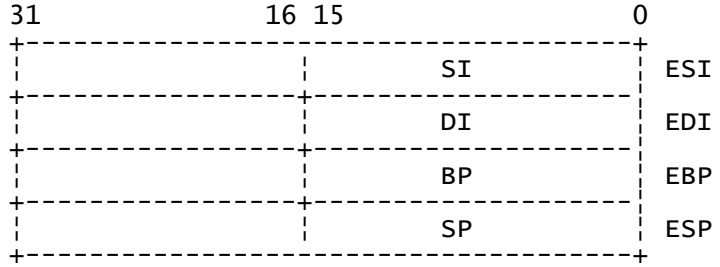
```
+----------------+
¦ ASSEMBLY XVIII ¦
+----------------+
```

Hummmm... Estamos na era dos 32 bits... então por que esperar mais para discutirmos as novidades da linha 386 e 486? Eles não diferem muito do irmão menor: o 8086. A não ser pelo fato de serem "maiores". :)

O 8086 e 80286 têm barramento de dados de 16 bits de tamanho enquanto o 386 e o 486 tem de 32 bits. Nada mais justo que existam modificações nos registradores também:



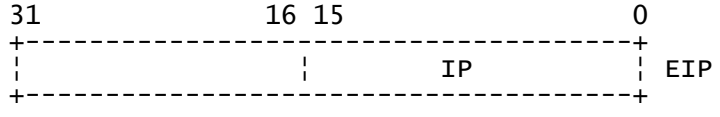
Os registradores de uso geral continuam os velhos conhecidos de sempre... Só que existem os registradores de uso geral de 32 bits: EAX, EBX, ECX e EDX, onde os 16 bits menos significativos destes são AX, BX, CX e DX, respectivamente.



Da mesma forma, os registradores SI, DI, BP e SP ainda estão aqui... bem como os seus equivalentes de 32 bits: ESI, EDI, EBP e ESP.

Os registradores de segmento (chamados de SELETORES desde o surgimento do 80286) são os mesmos e não mudaram de tamanho, continuam com 16 bits: CS, DS, ES e SS. Mas acrecentaram outros: FS e GS. Isto é... Agora existe um registrador de segmento de código (CS), um segmento de pilha (SS) e quatro segmentos de dados (DS, ES, FS e GS). Lembrando que DS é o segmento de dados default. Repare na ordem alfabética dos registradores de segmento de dados...

O registrador Instruction Pointer também continua o mesmo... E também existe o seu irmão maior... EIP:



Da mesma forma os FLAGS também são os mesmos de sempre... mas o registrador FLAGS também foi expandido para 32 bits e chamado de EFLAGS. Os sinalizadores extras são usados em aplicações especiais (como por exemplo, chaveamento para modo protegido, modo virtual,

chaveamento de tarefas, etc...).

Alguns outros registradores foram adicionados ao conjunto: CR0, CR1, CR3, TR4 a TR7. DR0 a DR3, DR6 e DR7 (todos de 32 bits de tamanho). Esses novos registradores são usados no controle da CPU (CR?), em testes (TR?) e DEBUG (DR?). Não tenho maiores informaçöes sobre alguns deles e por isso não vou descrevê-los aqui.

Novas instruçöes foram criadas para o 386 e ainda outras mais novas para o 486 (imagino que devam existir outras instruçöes específicas para o Pentium!). Eis algumas delas:

¦ BSF (Bit Scan Forward)

Processador: 386 ou superior

Sintaxe: BSF dest,src

Descrição:

Procura pelo primeiro bit setado no operando "src". Se encontrar, coloca o numero do bit no operando "dest" e seta o flag Zero. Se näo encontrar, näo altera o operando "dest" e reseta o flag Zero. BSF procura o bit setado começando pelo bit 0 do operando "src".

Exemplo:

```
BSF     AX,BX
```

¦ BSR (Bit Scan Reverse)

Processador: 386 ou superior

Sintaxe: BSR dest,src

Descrição:

Faz a mesma coisa que BSF, porém a ordem de procura começa a partir do bit mais significativo do operando "src".

Exemplo:

```
BSR     AX,BX
```

¦ BSWAP

Processador: 486 ou superior

Sintaxe: BSWAP reg32

Descrição:

Inverte a ordem das words de um registrador de 32 bits.

Exemplo:

```
BSWAP EAX
```

¦ BT (Bit Test)

Processador: 386 ou superior

Sintaxe: BT dest,src

Descrição:

Copia o conteúdo do bit do operando "dest" indicado pelo operando "src" para o flag Carry.

Exemplo:

```
BT  AX,3
```

Observações:

1- Aparentemente esta instrução não aceita operandos de 32 bits.
2- No exemplo acima o bit 3 de AX será copiado para o flag Carry.

¦ BTC (Bit Test And Complement)

Processador: 386 ou superior

Sintaxe: BTC dest,src

Descrição:

Instrução identica à BT, porém complementa (inverte) o bit do operando "dest".

¦ BTR e BTS

Processador: 386 ou superior

Sintaxe: BTR dest,src
BTS dest,src

Descrição:

Instruções identicas a BT, porém BTR zera o bit do operando destino e BTS seta o bit do operando destino.

¦ CDQ (Convert DoubleWord to QuadWord)

Processador: 386 ou superior

Sintaxe: CDQ

Descrição:

Expande o conteúdo do registrador EAX para o par EDX e EAX, preenchendo com o bit 31 de EAX os bits de EDX (extensão de sinal).

¦ CWDE (Convert Word to DoubleWord Extended)

Processador: 386 ou superior

Sintaxe: CWDE

Descrição:

Esta instrução expande o registrador AX para EAX, considerando o sinal. Ela é equivalente a instrução CWD, porém não usa o par DX:AX para isso.

¦ CMPXCHG

Processador: 486 ou superior

Sintaxe: CMPXCHG dest,src

Descrição:

Compara o acumulador (AL, AX ou EAX - dependendo dos operandos) com o operando "dest". Se forem iguais o acumulador é carregado com o conteúdo de "dest", caso contrário com o conteúdo de "src".

Exemplo:

```
CMPXCHG BX,CX
```

¦ INVD (Invalidate Cache)

Processador: 486 ou superior

Sintaxe: INVD

Descrição:

Limpa o cache interno do processador.

¦ JECXZ

Processador: 386 ou superior

Observação: É identica a instrução JCXZ, porém o teste é feito no registrador extendido ECX (32 bits).

¦ LGS e LFS

Processador: 386 ou superior

Observação: Essas instruções säo identicas as instruções LDS e LES, porém trabalham com os novos registradores de segmento.

¦ MOVSX e MOVZX

Processador: 386 ou superior

Sintaxe: MOVSX dest,src
MOVZX dest,src

Descrição:

Instruções úteis quando queremos lidar com operandos de tamanhos diferentes. MOVZX move o conteúdo do operando "src" para "dest" (sendo que "src" deve ser menor que "dest") zerando os bits extras. MOVSX faz a mesma coisa, porém copiando o último bit de "src" nos bits extras de "dest" (conversão com sinal).

Exemplo:

* Usando instruções do 8086, para copiar AL para BX precisariamos fazer isto:

```
MOV     BL,AL
MOV     BH,0
```

* Usando MOVZX podemos simplesmente fazer:

```
                MOVZX   BX,AL
```

¦ Instrução condicional SET

    Processador: 386 ou superior

    Sintaxe: SET? dest
             (Onde ? é a condição...)

    Descrição:

        Poe 1 no operando destino se a condição for satisfeita. Caso contrário poe 0.

    Exemplo:

```
        SETNZ AX
        SETS  EBX
        SETZ  CL
```

¦ SHRD e SHLD (Double Precision Shift)

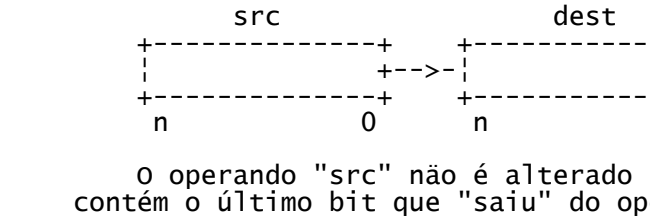
    Processador: 386 ou superior

    Sintaxe: SHRD dest,src,count
             SHLD dest,src,count

    Descrição:

        Faz o shift para esquerda (SHLD) ou direita (SHRD) do operando "dest" "count" vezes, porém os bits que seriam preenchidos com zeros são preenchidos com o contéudo dos bits do operando "src". Eis um gráfico exemplificando:

    SHRD



    O operando "src" não é alterado no processo. O flag de Carry contém o último bit que "saiu" do operando "dest".

    Exemplo:

```
        SHLD    EAX,ECX,3
        SHRD    AX,BX,CL
```

¦ Instruções que manipulam blocos...

    CMPSD, LODSD, MOVSD, STOSD, INSD e OUTSD se comportam da mesma forma que suas similares de 8 ou 16 bits (CMPSB, CMPSW, etc..), porém usam os registradores extendidos (ESI, EDI, ECX, EAX) e operam com dados de 32 bits de tamanho (DoubleWords).

    Existem mais instruções... Consulte algum manual da Intel ou o hipertexto HELPPC21... Pedirei aos Sysops do VixNET BBS (agora com 6 linhas hehehe) para deixarem disponivel o arquivo 386INTEL.ZIP... que é o guia técnico para o processador 386.

Dúvidas a respeito dos novos recursos:

¦Q¦ Os segmentos tem mais que 64k no modo real, já que os registradores extendidos podem ser usados neste modo? Como funcionaria uma instrução do tipo:

```
    MOV     [ESI+3],EAX
```

¦R¦ Näo... no modo real os segmentos continuam a ter 64k de tamanho. Os registradores extendidos podem ser usados a vontade e, quando usados como offset em um segmento, os 16 bits superiores säo ignorados. A instruçäo apresentada funcionaria da mesma forma que:

```
    MOV     [SI+3],EAX
```

¦Q¦ Onde e quando deve-se usar os novos registradores de segmentos?
¦R¦ Onde e quando você quiser. Pense neles como se fosse novos segmentos de dados extras. Na realidade você apenas conseguirá usá-los se explicitá-los numa instruçäo que faz referência à memória, por exemplo:

```
    MOV FS:[BX],AL
```

¦Q¦ Posso usar os registradores extendidos nas instruçöes normais ou apenas nas novas instruçöes?
¦R¦ Pode usá-los nas instruçöes "normais". A não ser que a instruçäo näo permita operandos de 32 bits...

That's all for now, pals...