1. Escreva um programa que exiba uma imagem disponibilizada com seus pixels brutos.

* Observações:
- O arquivo “avfinal.raw” (http://www.lia.ufc.br/~yuri/20181/arquitetura/avfinal.raw) contém os bytes brutos da imagem, isto é, cada byte representando cada pixel;
- A imagem possui 320x50 pixels (320 colunas e 50 linhas). Como cada pixel é representado por um byte, que, por sua vez, é o índice da cor na paleta de cores da placa de vídeo, o tamanho total da imagem é 16000 bytes. A imagem não contém uma paleta própria, então não há a necessidade de carregar tais dados;
- Para ser possível a montagem de um arquivo de imagem de disco para ser executado no boot da máquina, o arquivo deve possuir um tamanho igual a um número fechado de setores. Os 16000 bytes significariam 31,25 setores. Logo, são necessários, no mínimo, 32 setores para gravar o arquivo, o que implica em 16384 bytes. Esse, portanto, é o tamanho do “avfinal.raw”, sendo a diferença completada por bytes “0”;
- Você deve fazer a montagem do seu programa com o montador (“nasm”, de preferência) e depois concatenar o “avfinal.raw” ao arquivo binário gerado (no linux, use o comando “cat”).

* Dicas:
- No começo do seu programa, mude o modo de vídeo para o modo gráfico, usando a interrupção 0x10 (interrupção de vídeo) com o registrador AL em 13h, isto é, o modo VGA: 320x200 pixels, 256 cores, 1 byte por pixel;
- Utilize a interrupção 0x13 (interrupção de disco) para carregar os 32 setores referentes ao “avfinal.raw” (a imagem) na região de memória logo após o seu programa. Como o seu programa está no endereço 0x7C00 e possui 512 bytes de tamanho, o próximo endereço livre é 0x7E00. Carregue os setores nessa região;
- Finalmente, escreva um trecho de código para percorrer os 16000 bytes da imagem que estão no endereço 0x7E00, copiando-os para o endereço da memória ram de vídeo (VRAM), isto é, 0xA000. Como a imagem foi criada com a paleta padrão VGA, não há a necessidade de carregar uma nova paleta. Tais dados sequer existem no “avfinal.raw”, o qual contém exclusivamente os pixels.

2. Escreva um programa que decodifique uma mensagem criptografada em um arquivo texto.

*Observações:
- O arquivo texto criptografado é gerado no endereço http://www.lia.ufc.br/~yuri/20181/arquitetura/codifica.php . Entre com o seu número de matrícula, e o arquivo "msg.txt" será gerado;
- O arquivo "msg.txt" tem 512 bytes de tamanho, isto é, o tamanho de 1 setor, de tal forma que ele deve ser concatenado ao binário do seu programa (programa montado). No Linux, use o comando "cat" para isso;
- O arquivo usa uma criptografia bem simples: para cada caractere, pegue o seu byte e some o dígito correspondente de matrícula (desconsiderando os 0's à esquerda). Por exemplo, a mensagem:

```
Mensagem criptografada
```

Possui os bytes, de acordo com a tabela ASCII (mostrando em decimal):

```
77 101 110 115 97 103 101 109 32 99 114 105 112 116 111 103 114 97 102 97 100 97
```

Suponha que a sua matrícula seja 015236. Os dígitos 1 5 2 3 6 são usados, de forma circular, para gerar os bytes criptografados, que serão, neste exemplo:
77+1 = 78 ; 101+5 = 106 ; 110 + 2 = 112 ; 115 + 3 = 118 ; 97 + 6 = 103 ; 103 + 1 = 104 ; 101 + 5 = 106 ; 109 + 2 = 111 ; 32 + 3 = 35 ; 99 + 6 = 105 ; 114 + 1 = 115 ; 105 + 5 = 110 ; 112 + 2 = 114 ; 116 + 3 = 119 ; 111 + 6 = 117 ; 103 + 1 = 104 ; 114 + 5 = 119 ; 97 + 2 = 99 ; 102 + 3 = 105 ; 97 + 6 = 103 ; 100 + 1 = 101 ; 97 + 5 = 102
Ou seja:

```
78 106 112 118 103 104 106 111 35 105 115 110 114 119 117 104 119 99 105 103 101 102
```

Exibidos os caracteres correspondentes, a mensagem fica:

```
Njpvghjo#isnrwuhwcigef
```

*Dicas:
- Utilize a interrupção 0x13 (interrupção de disco) para carregar o setor que contém o texto na região de memória logo após o seu programa. Como o seu programa está no endereço 0x7C00 e possui 512 bytes de tamanho, o próximo endereço livre é 0x7E00. Carregue o segundo setor do disco nessa região;
- Coloque ao final do seu programa os bytes correspondentes aos dígitos de sua matrícula com a pseudo operação "db", dando um label para poder acessá-los mais facilmente, sem precisar se preocupar com o endereço de memória. Como sua matrícula pode conter o dígito 0 no meio dela, utilize o byte 10 como indicador de fim. Por exemplo, se sua matrícula é 0083504:

```
mat: db 8, 3, 5, 0, 4, 10
```

Perceba que não são necessárias aspas entre os dígitos da matrícula. Isso porque a criptografia é feita utilizando-se o byte do dígito propriamente dito, e não o byte do caractere que representa aquele dígito. Em outras palavras, 5 é o byte 5 mesmo, e não o byte 53 que é o caractere "5";
- Você pode copiar cada byte da sua matrícula para um registrador através de endereçamento direto da memória utilizando o label. Por exemplo:

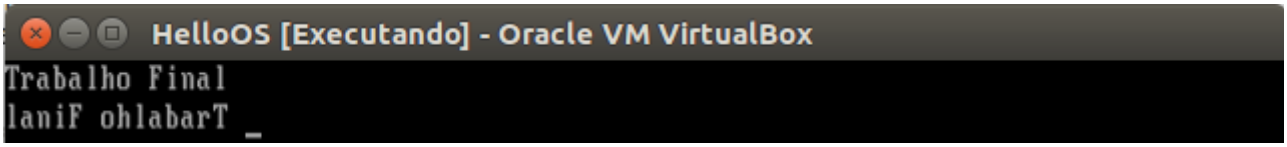
```
mov al, [mat]
```

Entretanto, para facilitar o acesso de cada byte, pode-se primeiro copiar "mat" (endereço de memória) para o registrador de índice si e depois acessar através de endereçamento de memória por registrador. Por exemplo:

```
mov si, mat
mov al, [si]
```

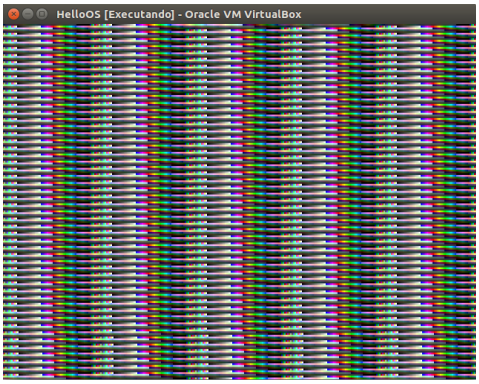
Dessa forma, você pode apenas incrementar o registrador SI para pegar o próximo byte. E, quando SI estiver com o endereço do primeiro byte após sua matrícula (byte 10), você tem como reatribuir o endereço do primeiro byte da matrícula, repetindo a cópia de mat para SI.

3. Faça um programa no qual o usuário digita algo e, ao apertar enter, o computador exibe o que foi digitado de trás para frente, de acordo com a imagem abaixo:



*Dicas:
- Utilize a interrupção 0x16 (interrupção de teclado) para pegar as teclas digitadas;
- Utilize a interrupção 0x10 (interrupção de vídeo), sendo AH com 0x0E, para exibir os caracteres digitados;
- O “Enter” emite o byte 13. Utilize-o para saber quando fazer a exibição do texto invertido;
- Para escrever em uma linha diferente o resultado, utilize a interrupção 0x10, sendo AH com 0x02 (atribuir posição do cursor) e DH com 1 (linha 1 – começa em 0). A interrupção 0x10, quando na função de escrita de caracteres, sempre exibe na posição do cursor.

4. Faça um programa que pinte toda a paleta de cores repetidamente na tela de forma animada, de forma que o resultado aparente uma televisão analógica sem sinal. O resultado estático é o seguinte:



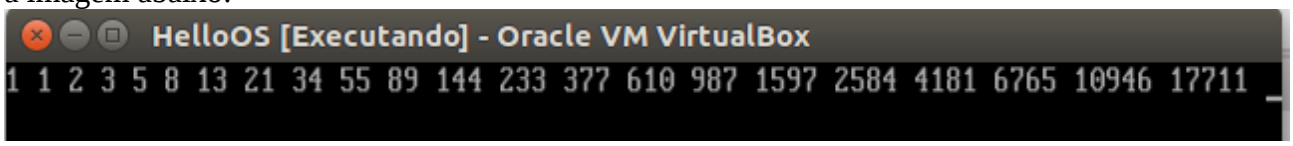
* Dicas:
- Utilize a interrupção 0x10 com AL = 0x13 para colocar no modo gráfico (VGA, 320 x 200, 256 cores);
- Para cada pixel, atribua a cor seguinte da paleta de cores. Faça isso acessando a memória ram de vídeo (VRAM), que está no endereço 0xA000;
- Para dar o efeito animado, sempre que pintar a tela completa, volte para o início da VRAM e repita o processo, porém iniciando numa cor diferente da paleta (a primeira pintura começa em 0. A segunda em 1…).

5. Escreva um programa que, em modo gráfico (VGA), a cada tecla apertada pelo usuário, a tela mude de cor para aquela correspondente ao byte da tecla digitada. Por exemplo, ao teclar ‘a’, o byte correspondente é 97. A cor 97, na paleta padrão VGA, por sua vez, é um verde claro.

* Observações:
- O programa deve ficar em loop infinito e mudar a cor da tela toda sempre que o usuário teclar uma nova tecla. Nada mais deve aparecer.

* Dicas:
- Utilize a interrupção 0x10 com AL = 0x13 para colocar no modo gráfico (VGA, 320 x 200, 256 cores);
- Utilize a interrupção 0x16 (interrupção de teclado) para pegar os bytes das teclas digitadas;
- Para cada pixel da tela, atribua o byte digitado. Faça isso acessando a memória RAM de vídeo (VRAM), que está no endereço 0xA000, percorrendo todas as 64000 posições (320x240). Quando terminar, espere uma nova tecla ser teclada.

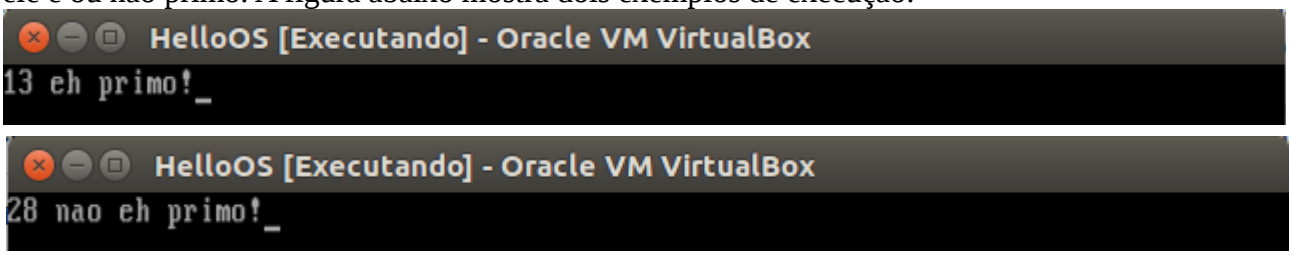
6. Faça um programa que exiba os 22 primeiros números da sequência de Fibonacci, de acordo com a imagem abaixo:



* Observação:
- Separe cada número por um espaço.

* Dicas:
- Comece escrevendo um procedimento para, dado um valor guardado no registrador AX, exibi-lo em tela, isto é, transforme nos caracteres correspondentes. Utilize a interrupção 0x10 (interrupção de vídeo) com AH = 0x0E para facilitar a escrita dos caracteres;
- Calcule cada número da sequência de Fibonacci em um laço (versão iterativa do algoritmo da sequência de Fibonacci), guardando o número da iteração atual em AX e usando o seu procedimento de exibição de valores para mostrar na tela.

7. Escreva um programa no qual o usuário digite um número, aperte enter, e o computador diga se ele é ou não primo. A figura abaixo mostra dois exemplos de execução:



* Observações:
- O programa apenas espera que o usuário digite o número. Ao apertar enter, o computador calcula e completa a linha com a mensagem correspondente, isto é, “ eh primo!” ou “ nao eh primo!”

* Dicas:
- Utilize a interrupção 0x16 (interrupção de teclado) para pegar os bytes das teclas digitadas;
- Utilize a interrupção 0x10 (interrupção de vídeo) com AH = 0x0E para facilitar a escrita dos caracteres;
- Crie um procedimento para transformar os caracteres digitados no número correspondente, ou seja, o algoritmo inverso ao demonstrado em sala de aula, que pegava um número guardado em um registrador e escrevia os caracteres correspondentes.