

Bem-vindo, visitante!

Login: Senha: Sempre ligado

Pesquisar neste tópico...

» Pesquisa avançada

Portal **Fórum** Downloads Resources Tutoriais Scripts Chat Membros Info**Para refletir:** "O maior desafio é sempre o último." - Tushin**Mundo RPG Maker - Fórum** » **Centro de Aprendizagem** » **Informática** » **Programação de Software (Moderador: Pititia12)** » **[Assembly] Como criar um sistema operacional - Aula 2**[Quer ser uploader da MRM? Carregue aq](#)Páginas: [1] [Ir para o fundo](#)Tópico: **[Assembly] Como criar um sistema operacional - Aula 2** (Lido 1314 vezes)

0 Membros e 1 Visitante estão vendo este tópico.

rodrigofms

Membro



Mensagens: 1007

Reputação: 119

OFFLINE**[Assembly] Como criar um sistema operacional - Aula 2**

« em: 24 de Março de 2011, 18:11:14 »

Como criar um sistema operacional em Assembly - Aula 2

Introdução

Bem, como planejado, aqui está a segunda aula de como criar um sistema operacional em Assembly. Se você seguiu corretamente a primeira aula, você deve estar com uma máquina virtual rodando o Ubuntu em seu computador. É importante que você esteja conseguindo acessar a Internet a partir de sua máquina virtual, além de estar conseguindo trabalhar confortavelmente nela. Caso você não esteja, provavelmente você não instalou as "Adições de Convidado" do VirtualBox. Ne caso, veja a [primeira aula](#). Se continuar com dificuldades, sinta-se a vontade para fazer perguntas.

O que aprenderemos nesta aula?

- O que é o NASM?
- O que é o QEMU?
- Instalando NASM e QEMU
- Escrevendo e rodando o código

O que é o NASM?

NASM significa **Netwide Assembler**.

É o programa que vamos usar para montar nosso código, ou seja, transformar nosso código em um binário que poderá ser executado.

O que é o QEMU?

QEMU é um emulador. Semelhante ao VirtualBox, ele emula um computador. Iremos usá-lo no Ubuntu para iniciar nosso sistema a partir da imagem de disquete que iremos criar.

Instalando o NASM e o QEMU

e um pacote de ferramentas essencial para o desenvolvimento

Instalar o NASM e o QEMU no Ubuntu é incrivelmente fácil.

Na barra do Ubuntu que fica no topo da tela, clique em Aplicativos > Acessórios > Terminal.

Um terminal irá abrir.

Para instalar o NASM e o QEMU, basta que você digite uma linha de código.

Nota: isso não vai funcionar se você não estiver conseguindo acessar a Internet a partir de sua máquina virtual.

Código: [\[Selecionar\]](#)

```
sudo apt-get install build-essential qemu nasm
```

Ele provavelmente irá pedir uma senha. É a senha do administrador que você escolheu quando instalou o Ubuntu. Depois disso, apenas espere o comando terminar de rodar. Quando acabar de rodar e o terminal estiver aguardando seu comando, você pode fechá-lo. O NASM e o QEMU já estão instalados.

Escrevendo o código

Agora, nossa máquina virtual está pronta para ser usada para a criação de nosso sistema operacional. Então vamos come

É recomendado que você crie uma nova pasta na área de trabalho do Ubuntu, onde vamos guardar os arquivos de nosso s
Nota: evite nomes "compostos", como "Meu sistema". Use uma palavra só.

Eu criei uma pasta chamada "MRM".

Por enquanto, nosso sistema simplesmente escreverá uma mensagem na tela.

Vamos começar a escrever o código. Abra o Editor de Texto, que está localizado em Aplicativos > Acessórios > Editor de Te

Escreva o seguinte código:

Código: [Selecionar]

```

BITS 16

mov ax, 07C0h
mov ss, ax
mov ds, ax

mov si, Msg
mov ah, 0Eh
jmp repetir

Msg db 'Bem-vindo ao MRM OS!', 0

repetir:
lodsb
cmp al, 0
je fim
int 10h

```

Sim, este é o código do nosso sistema operacional. Vamos entendê-lo?

A primeira linha diz **BITS 16**. Ela serve simplesmente para dizer ao **NASM** que estamos trabalhando em modo 16 Bits.

As três linhas de código seguintes simplesmente setam o segmento de pilha e o segmento de dados para 07C0h. Na verd
é necessário que você entenda isso muito bem. É simplesmente algo que precisa estar ali.

mov si, Msg copia a localização da mensagem que iremos exibir para o registro **SI**.

mov ah, 0Eh serve para configurar o valor no registro **AH**. Este valor serve para imprimir um caractere na tela.

jmp repetir pula para a parte do código depois de **repetir**. Assim, o computador não vai tentar executar **Msg db 'Bem vindo : OS',0**, que é a mensagem que vamos exibir na tela.

lodsb carrega um caractere da mensagem que vamos imprimir. **cmp al, 0** verifica se chegou ao 0 (em **'Bem-vindo ao MRM OS',0**). Se sim, isso significa que a mensagem acabou, então **je fim** pula para a parte **fim** do código. É como **jmp**, porém neste cas
executado se o resto todo da mensagem já estiver na tela.

int 10h acessa a função de imprimir caractere que configuramos antes com **mov ah, 0Eh**.

jmp repetir volta para o início da parte **repetir** do código.

Em **fim**, **jmp \$** cria um loop infinito. Este comando serve para que o código pule para o mesmo lugar, fazendo com que o pr
pare.

times 510-(\$-\$\$) db 0 serve para aumentar o arquivo binário com zeros até que ele tenha 510 bytes, e **dw 0xAA55** é a assir
boot.

Como assim?

Bem, para que um arquivo seja reconhecido como um arquivo válido para o computador ser iniciado, ele precisa ter exatam
bytes e terminar com a assinatura de boot, **0xAA55**.

O comando **times 510-(\$-\$\$) db 0** enche o arquivo até que ele tenha 510 bytes, e o comando **dw** no final é responsável pel
últimos 2kb.

Rodando o código

Salve o seu arquivo como "meuos.asm" na pasta que criou na Área de Trabalho.

Agora, volte ao Terminal.

Digite **cd Área\de\Trabalho** e depois **cd *nome da sua pasta aqui*** para navegar até a pasta que você criou e onde você sa
arquivo.

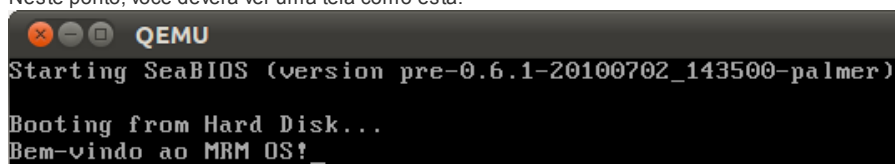
Agora, vamos transformá-lo em um binário.

Para isso, usaremos o NASM. Basta digitar o comando **nasm -o meus.bin meus.asm -f bin**.

Se tudo deu certo, você irá observar que um arquivo chamado **meuos.bin** apareceu na mesma pasta.

Para rodar o seu sistema operacional, vamos usar o QEMU. Ainda no terminal, digite **qemu meus.bin**.

Neste ponto, você deverá ver uma tela como esta:



```

QEMU
Starting SeaBIOS (version pre-0.6.1-20100702_143500-palmer)
Booting from Hard Disk...
Bem-vindo ao MRM OS!_

```

Na próxima aula...

Na próxima aula, veremos **como criar um CD com este pequeno sistema operacional**.

Você deve ter reparado também que eu disse "para que um arquivo seja reconhecido como um arquivo válido para o comp

ser iniciado, ele precisa ter exatamente **512 bytes** e terminar com a assinatura de boot". Ou seja, se quisermos criar um sistema que pese mais que **512 bytes**, teremos que fazer de outra forma.

Espero que tenham gostado desta aula, e até a próxima. 

[Ir para a próxima aula >>](#)

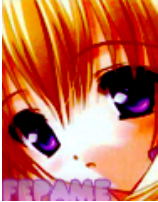
« Última modificação: 24 de Abril de 2011, 21:39:51 por rodrigofms »



rodrigofms has fled the battle.

Fepame

Membro



Mensagens: 113
Reputação: 52

OFFLINE



Re: [Assembly] Como criar um sistema operacional - Aula 2

Pontuação

« Resposta #1 em: 24 de Março de 2011, 18:14:43 »

Que feio, prefiro meu Sistema Fenix1

Estou brincando, parabéns, vejo um novo steve jobs vindo ae :D
~~E me contratando futuramente re~~



Lumina Fox

Membro



Mensagens: 19
Reputação: 2


OFFLINE



Re: [Assembly] Como criar um sistema operacional - Aula 2

Pontuação

« Resposta #2 em: 24 de Março de 2011, 18:38:08 »

"He's a god of programming, Wright, a god!" 

Enfim, vamos às críticas quanto ao tutorial e depois aos elogios e afins. _.

Código: [Selecionar]

```
mov ax, 07C0h
mov ss, ax
mov ds, ax
```

Entendo que não seja necessário entender, mas seria bom você explicar isso. Eu não sei por que, apenas acho que seria bom uma informação a mais.

Mas mesmo assim, Rod, você sabe que eu não tenho jeito para essas coisas de programação, então não pude formar algo concreto. Apenas li, pensei, comentei. e.e"


<logo de elogios vem aqui>

Você sabe que acho você um deus da programação, mesmo que existam tantas pessoas melhores que você. _.

E "deus" não seja um termo muito certo, mas deu para entender, né? D:

Mas se for ver, essas pessoas tem entre 17/18 anos. E você tem apenas 13/14. o.o Você vê a seriedade disso?

Se você já é o que é agora, imagine depois. =)

Mas enfim, a aula ficou explicativa, gostei. 

E como o Fepame disse acima, se um dia você for um Steve Jobs da vida, me sustente, please. '!

Não estou brincando. =P

Noite o/



rodrigofms

Membro



Mensagens: 1007
Reputação: 119

OFFLINE



Re: [Assembly] Como criar um sistema operacional - Aula 2

Pontuação

« Resposta #3 em: 24 de Março de 2011, 18:42:41 »

Citação de: Lumina Fox em 24 de Março de 2011, 18:38:08

Código: [Selecionar]

```
mov ax, 07C0h
mov ss, ax
mov ds, ax
```

Entendo que não seja necessário entender, mas seria bom você explicar isso. Eu não sei por que, apenas acho que seria bom, é uma informação a mais.

Eu expliquei como dava pra explicar. 

Citação de: rodrigofms em 24 de Março de 2011, 18:11:14

As três linhas de código seguintes simplesmente setam o segmento de pilha e o segmento de dados para 07C0h. Na verdade, não é necessário que você entenda isso muito bem. É simplesmente algo que precisa estar ali.

Em outras palavras, servem para configurar o segmento de pilha e segmento de dados.

E obrigado pelos elogios. 



rodrigofms has fled the battle.

João B

Membro



Mensagens: 75
Reputação: 330

OFFLINE



Re: [Assembly] Como criar um sistema operacional - Aula 2

« Resposta #4 em: 24 de Abril de 2011, 16:39:35 »

Pontuaç

Citação de: rodrigofms em 24 de Março de 2011, 18:42:41

Citação de: Lumina Fox em 24 de Março de 2011, 18:38:08

Código: [Selecionar]

```
mov ax, 07C0h
mov ss, ax
mov ds, ax
```


Entendo que não seja necessário entender, mas seria bom você explicar isso. Eu não sei por que, apenas acho que seria bom, é uma informação a mais.

Eu expliquei como dava pra explicar. 

Citação de: rodrigofms em 24 de Março de 2011, 18:11:14

As três linhas de código seguintes simplesmente setam o segmento de pilha e o segmento de dados para 07C0h. Na verdade, não é necessário que você entenda isso muito bem. É simplesmente algo que precisa estar ali.

Em outras palavras, servem para configurar o segmento de pilha e segmento de dados.

E obrigado pelos elogios. 

Só complementando... o segmento de "dados" seria aonde as instruções **mov** e afins teriam efeitos, e o segmento da "pilha" aonde instruções **push** e **pop** teriam efeito.

Ah, e você se esqueceu de iniciar o segmento extra **es**, porque, que eu saiba, as instruções **stos*** usam esse segmento.



Modificando o Banco de Dados Dinamicamente com RGSS

Alguém pode fixar o tópico acima, por favor?

rodrigofms

Membro



Mensagens: 1007
Reputação: 119

OFFLINE



Re: [Assembly] Como criar um sistema operacional - Aula 2

« Resposta #5 em: 24 de Abril de 2011, 21:40:11 »

Pontuaç

Tópico atualizado com link para a próxima aula.



rodrigofms has fled the battle.




Páginas: [1] [Ir para o topo](#)

« anterior

[Mundo RPG Maker - Fórum](#) » [Centro de Aprendizagem](#) » [Informática](#) » [Programação de Software \(Moderador: Pititia12\)](#) » [\[Assembly\] Como criar um sistema operacional - Aula 2](#)

Ir para:

Tópicos Relacionados

	Assunto / Iniciado por	Respostas	Última mensagem
	Novo sistema operacional Iniciado por Bee50cent no quadro Computadores: Ajuda & Discussão	1 Respostas 323 Visualizações	28 de Agosto de 2010, 12:44 por Johnny Mercy
	[Assembly] Como criar um sistema operacional - Aula 1 Iniciado por rodrigofms no quadro Programação de Software	15 Respostas 1808 Visualizações	09 de Julho de 2011, 18:45:00 por gmsaldanha
	[Assembly] Como criar um sistema operacional - Aula 3 Iniciado por rodrigofms no quadro Programação de Software	1 Respostas 986 Visualizações	25 de Abril de 2011, 12:30:00 por rafaelmelo

Powered by SMF 2.0 | SMF © 2006–2011, Simple Machines LLC
© Mundo RPG Maker - Alguns Direitos Reservados
Layout por Paoa

Volt