UNIVERSIDADE DE BRASÍLIA

Tutorial para o trabalho de Software Básico 1º/2011

Como testar seu programa

Ana Carolina Cardoso de Sousa
Murilo Marques Marinho

30 de Abril de 2011

Resumo de como utilizar o Desmontador da Ana para testar seus objetos e executáveis montados pelo Montarilo e dicas para o desenvolvimento do trabalho

Conteúdo

1. Introdução 3
2. Montando seus programas em 8085 3
3. Desmontando seus programas em 8085 3
3.1. Desmontando um arquivo objeto (1ª parte do trabalho) 4
3.2. Desmontando um arquivo executável (2ª parte do trabalho) 4
4. Dicas para o desenvolvimento do trabalho 5

1. Introdução

Na primeira parte do trabalho, vocês terão que fazer um novo pré-processamento para o programa já funcional do Murilo. Mas como a saída é do tipo objeto (.o), é muito trabalhoso verificar só pelos métodos tradicionais (utilizando o *objdump* ou o *readelf*). Para poupar um pouco esse trabalho, a idéia é utilizar o meu desmontador.

O mesmo vale para a segunda parte do trabalho, aonde vocês terão que criar um ligador, obtendo saídas executáveis do tipo elf (.elf).

Quando fiz o programa, tentei abordar todas as possibilidades possíveis de arquivos de entrada. Mas eu não acharia estranho vocês acharem problemas nele conforme forem o utilizando. Paciência... Postem possíveis problemas/dúvidas no fórum.

Alterei algumas coisinhas no meu programa esse fim de semana, então talvez os fluxogramas não estejam 100% corretos. Mas nem faz tanta diferença, já que vocês não vão mexer no meu programa, só utilizá-lo.

Lembrando que um desmontador não deveria poder ler um arquivo objeto (que só possui o Section Header), ele só desmonta arquivos executáveis (que possui o Program Header e pode ou não possuir o Section Header[1]). Adaptei-o para vocês poderem utilizá-lo nas duas partes do trabalho.

2. Montando seus programas em 8085

Comece compilando o *Montarilo*. Na pasta aonde você extraiu todos os arquivos do *aprender*, escreva no terminal:

>> make

Para montar os arquivos exemplos que o Murilo criou e deixou na pasta *exemplos* escreva:

>> make exemplos

Esses exemplos serão montados na pasta *exemplos_montados*. Observe que eles serão todos arquivos objetos (.o).

Para montar seus próprios exemplos, utilize a seguinte linha de comando:

>> ./montador arquivodeentrada.txt arquivodesaida

Para maiores informações sobre quais comandos existem no *makefile* e sobre o que o programa do Murilo respeita como entrada de arquivos, leia o *README* do *Montador v3.*1.

3. Desmontando seus programas em 8085

[1] Isso é parte da matéria que vocês devem aprender em breve. Sem desesperos...

Agora para compilar o meu desmontador é exatamente o mesmo como acima, na pasta aonde você extraiu os arquivos do *aprender*, escreva no terminal:

>> *make*

Para maiores informações sobre quais comandos existem no *makefile* e sobre o que o meu programa respeita como entrada de arquivos, leia o *README* do *Desmontador v3.*1.

3.1. Desmontando um arquivo objeto (1ª parte do trabalho)

Copie os arquivos que você montou acima na pasta *exemplos*. O meu comando para testar esses exemplos é:

>> *make desmontar*[2]

Esses exemplos serão desmontados na pasta *exemplos_desmontados*.

Para desmontar seus próprios exemplos, utilize uma das seguintes linhas de comando:

>> *./desmontador -g arquivomontado desmontado.txt*
>> *./desmontador -g arquivomontado*

No primeiro caso o *arquivomontado* será desmontado em um arquivo texto chamado *desmontado.txt*.

No segundo caso o *arquivomontado* será desmontado em um arquivo texto de mesmo nome, no caso *arquivomontado.txt*.

Note esse *–g*. É o comando para o meu programa entender que você está tentando desmontar um arquivo objeto que possui apenas o Section Header.

Mas mesmo que você por acaso se esqueça de escrever esse comando, meu programa apenas dará uma mensagem dizendo que não é possível desmontar e indicará a forma correta de utilizá-lo (com o *–g*).

Nesse modo existe ainda a opção de ver a saída na tela, para poupar o trabalho de ficar olhando os arquivos no editor de texto. Apenas aceite escrevendo *s*. Ou, se você não quiser ver a saída na tela, apenas digite qualquer outra tecla.

3.2. Desmontando um arquivo executável (2ª parte do trabalho)

Nessa parte do trabalho, vocês já estarão testando a saída dos ligadores que vocês criaram. Então, estaremos lidando apenas com arquivos executáveis.

[2] Confesso que não consegui deixar o comando igual ao do murilo "make exemplos", não sei porque não dá certo. Se alguém conseguir descobrir porque o makefile não deixa eu por isso e quiser me contar, vale pelo conhecimento adquirido. Mas liga pra isso não, é só preciosismo meu...

Não criei um script para arquivos executáveis, nem deixei nenhum como exemplo. Quando vocês estiverem nessa parte do trabalho serão capazes já de criar esse script no meu *makefile* e os exemplos serão suas próprias saídas.

Agora os comandos pra desmontar arquivos executáveis são:

>> *./desmontador arquivomontado desmontado.txt*
>> *./desmontador arquivomontado*

No primeiro caso o *arquivomontado* será desmontado em um arquivo texto chamado *desmontado.txt*.

No segundo caso o *arquivomontado* será desmontado em um arquivo texto de mesmo nome, no caso *arquivomontado.txt*.

Note a ausência de *–g*. Isso significa que o meu programa vai ler apenas pelo Program Header.

Caso você tente colocar o *–g* e não existir um *Section Header*, meu programa dará uma mensagem dizendo que não existe esse cabeçalho, então não o desmontará.

4. Dicas para o desenvolvimento do trabalho

Conversei um pouco com o Djore outro dia e eu acabei indicando várias dicas para ele de como fazer o trabalho, então vou deixar aqui para todo mundo também.

- Faça backups. Alterou uma função e funcionou? Zipa todos seus arquivos e salva como uma nova versão: "desmontador (v2.0.3)".
  - Faça um arquivo texto, indicando o que foi alterado em cada versão.
- Faça com antecedência. Eu sei que é de praxe pra deixar pro final. Mas a questão é que não dá tempo de fazer em uma semana... Só pra entender o que o Murilo fez já vai levar um bom tempo (e olha que ele é um garoto organizado).
- Usem o que vocês aprenderam no pré-processamento para fazerem partes do seu código só para Debug. Se vocês abrirem alguns dos meus arquivos, notarão que várias partes estão entre *#ifdef x* e *#endif*. Meu programa estava dando problema ontem com um detalhezinho, foi mil vezes mais fácil achar onde estava o erro só porque eu tinha deixado essas opções de Debug.
- Primeira ação a se fazer no inicio desse trabalho é abrir o *makefile* dele e entender o que está acontecendo. Vocês vão utilizá-lo muito.
  - Depois que você abriu e tem uma idéia do que está acontecendo, tenta criar um arquivo com uma nova função do estilo *HelloWorld* e colocar em alguma parte do trabalho a chamada para essa função. Se você souber fazer isso, provavelmente vai conseguir **começar** a fazer o trabalho.
- Para a segunda parte do trabalho, vocês passarão muito tempo aprendendo a ler arquivos elf. Os programas padrões de leitura desse tipo de programa são: *readelf* e *objdump*. Como eu tive que fazer os meus próprios programas executáveis, justamente por não ter um ligador, utilizei muito o *Bless* (*apt-get install bless*). Com ele é possível ver o arquivo sob uma outra perspectiva

(por bytes alinhados) e é possível editar os arquivos. Se vocês acharem útil para algo, utilize-o.

- Ás vezes vocês vão precisar de bibliotecas específicas. O melhor lugar, na minha opinião, para saber como elas funcionam, quais argumentos de entrada, saída, etc. é aqui: http://www.cplusplus.com/reference/clibrary/cstdio/printf/

Enfim, é isso. Divirtam-se.