Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Sistemas embarcados – 2021/1

Trabalho Prático 3 - Ligador

1. Descrição Geral

O principal objetivo deste trabalho é implementar um ligador para a máquina especificada, a ser executada no simulador CPUSim, ambos disponibilizados com este documento.

O CPUSim é desenvolvido em linguagem Java, e é capaz de simular uma máquina completa. Ele recebe como entrada um arquivo contendo os dados em linguagem de máquina, o qual é carregado na memória RAM da máquina especificada, deixando-o pronto para execução.

Mais informações sobre o CPUSim podem ser encontradas na página oficial do simulador, em http://www.cs.colby.edu/djskrien/CPUSim/.

2. Informações Importantes

- O trabalho deve ser feito em duplas, podendo ser discutido entre os colegas desde que não haja cópia ou compartilhamento do código fonte.
- A data de entrega será especificada através de uma tarefa no Moodle.
- Os trabalhos poderão ser entregues até às 23:55 do dia especificado para a entrega. O horário de entrega deve respeitar o relógio do sistema Moodle. Haverá uma tolerância de 5 minutos de atraso, de forma que os alunos podem fazer a entrega até às 0:00. A partir desse horário, os trabalhos já estarão sujeitos a penalidades. A fórmula para desconto por atraso na entrega do trabalho prático é:

  $$Desconto = 2^d/0.32\ \%$$

  onde *d* é o atraso em dias úteis. Note que após 5 dias úteis, o trabalho não pode ser mais entregue.
- O trabalho deve ser implementado obrigatoriamente nas linguagens C, C++ ou Python.
- Deverá ser entregue o código fonte com os arquivos de dados necessários para a execução e um arquivo *Makefile* que permita a compilação do programa nas máquinas Linux do DCC.
- Além disso, deverá ser entregue uma pequena documentação contendo todas as decisões de projeto que foram tomadas durante a implementação, sobre aspectos não contemplados na especificação, assim como uma justificativa para essas decisões. Esse documento não precisa ser extenso (mínimo 3 e máximo de 6 páginas). A documentação deve indicar o nome dos alunos integrantes do grupo.

- A ênfase do trabalho está no funcionamento do sistema e não em aspectos de programação ou interface com o usuário. Assim, não deverá haver tratamento de erros no programa de entrada.
- Todas as dúvidas referentes ao trabalho serão esclarecidas por meio do fórum disponível no ambiente Moodle da disciplina.
- A entrega do trabalho deverá ser realizada pelo Moodle, na tarefa criada especificamente para tal. A entrega deverá ser feita no seguinte formato:
    - O trabalho a ser entregue deverá estar contido em um único arquivo compactado, em formato "*.zip*".
    - O arquivo .zip definido deverá ter três pastas:
        - "*assembler*": Essa pasta deverá conter o código-fonte do montador implementado (entregue no TP2), mais o código do ligador, juntamente do arquivo *Makefile* (OBS.: Não dever'ao ser incluídos arquivos .o nem executáveis nessa pasta.)
        - "*tst*": Essa pasta irá conter os arquivos ".a" de entrada desenvolvidos para testar o montador e o ligador. Mais detalhes desse arquivo estão definidos na Seção 5.
        - "*doc*": Essa pasta deverá conter o arquivo da documentação, em formato PDF. Caso o grupo julgue necessário incluir quaisquer outros arquivos a parte, esses deverão ser justificados em um arquivo texto com o nome README.
    - A integridade do arquivo submetido no sistema é de inteira responsabilidade do aluno, ou seja, trabalhos com arquivos corrompidos terão nota zero.
    - **Atenção**: Trabalhos que descumprirem o padrão definido acima serão penalizados.

3. Especificação do ligador

O ligador que será desenvolvido deve suportar a ligação com símbolos externos aos módulos. Assim, vamos adicionar mais quatro palavras reservadas à linguagem de montagem suportada pelo montador desenvolvido anteriormente:

- **`.externD`** – Usado para indicar que o módulo corrente usa um dado externo ao módulo (D se refere a Data)
- **`.externT`** – Usado para indicar que o módulo corrente usa um procedimento externo ao módulo (T se refere a Text)
- **`.globalD`** – Indica ao ligador que o dado que o sucede poderá vir a ser usado por outros módulos do programa
- **`.globalD`** – Indica ao ligador que o procedimento que o sucede poderá vir a ser usado por outros módulos do programa

Vejam, então, que será necessário modificar o montador feito anteriormente para que ele gere uma tabela de relocação, que será utilizada pelo montador. Esta tabela de relocação, geralmente, é uma

seção do arquivo-objeto que é a saída do montador. Para facilitar, propomos que a tabela de relocação seja um arquivo em separado, com o sufixo ".sym". Assim, quando compilamos o programa "complexo.a", seria gerado o programa montado (ou complexo.hex ou complexo.mif), mais a tabela de relocação, com o nome complexo.sym.

O formato da tabela de relocação é livre. Por isso, espera-se que a documentação apresente, em alto nível, quais tipos de informação são exportados para o arquivo.sym e qual é o seu formato. Outra coisa a ser citada na documentação é a sua estratégia de alocação de símbolos: como o ligador desenvolvido organiza os dados e o texto na memória? A título de exemplo, poderíamos colocar os dados antes da pilha, ou poderíamos colocar os dados na estratégia *heap*-pilha, onde a pilha cresce para um sentido e o *heap* cresce para outro sentido. Outras opções ainda são possíveis.

O ligador é chamado indicando o nome dos módulos a serem ligados, mais o nome do executável final. Para simplificar, vamos assumir sempre que o último parâmetro é o nome do programa ligado. Por exemplo:

```
ligador complexo.hex main.hex ligado.hex
```

Essa chamada irá ligar os módulos *complexo* e *main*, gerando o executável *ligado* (no exemplo estamos supondo o formato hex como saída, mas lembre-se que o formato mif também pode ser escolhido).

Para fins de simplificação da especificação e do trabalho, estamos assumindo aqui que todos os símbolos exportados por módulos ligados devem ser considerados no momento de ligar o executável. Isso gera duas consequências:

1. **Consequência boa:** eliminamos uma nova palavra-chave reservada na linguagem de montagem. Geralmente temos o `.include` ou algo similar, que indica onde podemos encontrar os símbolos não definidos internamente.
2. **Consequência ruim:** O símbolo a ser procurado para resolver as dependências pode estar em qualquer outro módulo do programa. Por exemplo, se tivermos 4 módulos (A, B, C e D), para terminar a ligação de A devemos procurar na tabela de relocação de B, C e D pelos símbolos externos a A.

4. Sobre a Documentação

- Deve conter todas as decisões de projeto.
- Deve conter informações sobre cada programa testado, sobre o que ele faz, entradas de dados, saída esperada, etc.
- Deve conter elementos que comprovem que o montador e o ligador foram testados (Ex.: imagens das telas de montagem e execução no CPUSim). Quaisquer arquivos relativos a testes devem ser enviados no pacote do trabalho, como mencionado na Seção 2. A documentação deve conter referências a esses arquivos, explicação do que eles fazem e dos resultados obtidos.
- O código fonte não deve ser incluído no arquivo PDF da documentação.

5. Considerações Finais

É obrigatório o cumprimento fiel de todas as especificações descritas neste documento. As decisões de projeto devem fazer parte apenas da estrutura interna do montador, não podendo afetar a interface de entrada e saída.