Insper

São Paulo, 19/04/17

Elementos de Sistemas 2017.1

Sprint 6 – Assembler

Prazo de entrega: 3/04/17

Descrição:

A entrega dessa *sprint* é um Assemblers para a arquitetura Z0. O Assembler devera ser capaz de gerar código de máquina para qualquer arquivo de entrada válido em Assembly do Z0. O desenvolvimento será na linguagem Java, devem passar em todos os testes na integração contínua.

Organização:

A equipe deverá se organizar para que todos os módulos necessários sejam desenvolvidos e devidamente integrados no repositório master no Github do projeto, atualizem o fork. Cada tarefa é para ser realizada em grupos de dois participantes. Todos devem participar do desenvolvimento, embora só um integrante da dupla possa fazer os *commits* e *pull requests* para o Github. Não aceite código pronto de outros.

O facilitador escolhido é responsável pela organização da equipe e pela completude e consistência do *branch master* do fork do grupo Ele deverá chegar num consenso com a equipe para datas de entregas parciais. Além disso deverá conduzir, no encontro seguinte ao prazo de entrega, reuniões de Revisão e Retrospectiva.

Entrega:

As entregas dos grupos serão consideradas concluídas quando os *pullrequests* forem feitos e a tarefa tenha sido movida para DONE no Trello. O facilitador terá de verificar se tudo funciona corretamente e emitir um relatório no Trello caso negativo, garantindo que a *sprint* do kanban foi fechada.

Rua Quatá, 300 – Vila Olímpia 04546-042 São Paulo SP Brasil
55 11 45042400 www.insper.edu.br

Instruções:

O Assembler é um programa que carrega um arquivo texto em ASCII com as instruções em Assembly (extensão .nasm) e deve retornar um arquivo também em texto com as instruções de máquina no padrão MIF. Desenvolva o código na estrutura do diretório do `Assembler`, e após validar seu desenvolvimento, submeta um *pull request* com suas contribuições para que o facilitador do projeto aprove e faça o *merge* no *branch master*. O título do *pull requeste* deve ser o nome dos módulos desenvolvidos para facilitar a identificação do mesmo durante os merges. Não há necessidade de colocar os nomes de quem implementou, eles já estão no controle de versão e no Trello.

Módulos:

Devem ser desenvolvidos em Java:

- **Parser**: Desmembra cada comando em seus campos;
- **Code**: Traduz cada campo em seu valor binário;
- **SimbolTable**: Gerencia a tabela de símbolos;
- **AssemblerZ0**: Inicia leitura e escrita dos arquivos e controla aplicativo;

Links:

- https://github.com/ElementosDeSistemas/Z0
- Trello: https://trello.com/engcompinsper2017

Rua Quatá, 300 – Vila Olímpia 04546-042 São Paulo SP Brasil
55 11 45042400 www.insper.edu.br

Estrutura de Classes :

Parser: Encapsula o código de leitura. Carrega as instruções na linguagem assembly, analisa, e oferece acesso as partes da instrução (campos e símbolos). Além disso, remove todos os espaços em branco e comentários.			
Rotina	***Argumentos***	***Retorno***	***Função***
"construtor"	String	---	Abre o arquivo de entrada NASM e se prepara para analisá-lo.
advance	---	Boolean	Carrega uma instrução e avança seu apontador interno para o próximo linha do arquivo de entrada. Caso não haja mais linhas no arquivo de entrada o método retorna "Falso", senão retorna "Verdadeiro".
command		String	Retorna o comando "intrução" atual (sem o avanço)
commandType	String	A_COMMAND, C_COMMAND, L_COMMAND,	Retorna o tipo da instrução passada no argumento: A_COMMAND para leaw, por exemplo leaw $1,%A L_COMMAND para labels, por ex. Xyz: , onde Xyz é um símbolo. C_COMMAND para todos os outros comandos
symbol	String	String	Retorna o símbolo ou valor numérico da instrução passada no argumento. Deve ser chamado somente quando commandType() é A_COMMAND.
label	String	String	Retorna o símbolo da instrução passada no argumento. Deve ser chamado somente quando commandType() é L_COMMAND.
instruction	String	String[]	Separa os mnemônicos da instrução fornecida em tokens em um vetor de Strings. Deve ser chamado somente quando CommandType () é C_COMMAND.

Code: Traduz mnemônicos da linguagem assembly para códigos binários da arquitetura Z0.			
Rotina	**Argumentos**	**Retorno**	**Função**
dest	String[]	3 bits	Retorna o código binário do(s) registrador(es) que vão receber o valor da instrução.
comp	String[]	7 bits	Retorna o código binário do mnemônico para realizar uma operação de cálculo.
jump	String[]	3 bits	Retorna o código binário do mnemônico para realizar uma operação de jump (salto).
toBinary	String	15 bits	Retorna o código binário de um valor decimal armazenado numa String.

SymbolTable: Mantém uma tabela com a correspondência entre os rótulos simbólicos e endereços numéricos de memória.			
Rotina	**Argumentos**	**Retorno**	**Função**
"construtor"	---	---	Cria uma tabela de símbolos.
addEntry	String , int	---	Insere uma entrada de um símbolo com seu endereço numérico na tabela de símbolos.
contains	String	Boolean	Confere se o símbolo informado já foi inserido na tabela de símbolos.
getAddress	String	int	Retorna o valor numérico associado a um símbolo já inserido na tabela de símbolos.

AssemblerZ0: Classe principal que orquestra execução do Assembler.		
Parametro	**Argumentos**	**Função**
arquivo	*<arquivo nasm>*	primeiro parâmetro é o nome do arquivo nasm a ser aberto
-f	*<arquivo mif>*	parâmetro -f <arquivo> indica onde será salvo o arquivo gerado .mif

Rua Quatá, 300 – Vila Olímpia 04546-042 São Paulo SP Brasil
55 11 45042400 www.insper.edu.br