Insper

Elementos de Sistema - Projeto H - Assembler

Rafael Corsi - rafael.corsi@insper.edu.br

Abril - 2018

> **Prazo original : Terça Feira - 01/5/2018**
> **Atrasado até : Terça Feira - 08/5/2018**

Descrição

Nesse projeto iremos criar o programa *assembler* que é responsável por traduzir os códigos escrito em Assembly para a linguagem de máquina.

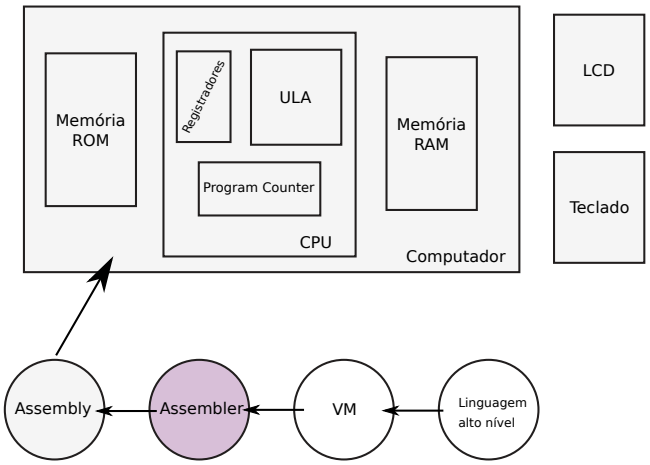

Figure 1: Assembly

Instruções

As instruções técnicas referente ao projeto estão no documento H-Ferramental.pdf

Módulos a serem implementados

O projeto no total possui 5 módulos, sendo que o módulo AssemblerZ01.java que já foi entregue implementando.

> Os módulos estão listados de maneira Top - Down

- AssemblerZ01
  - **Arquivo** : AssemblerZ01.java
  - **Descrição** : Main do projeto. Recebe como parâmetro o nome do arquivo .nasm e o nome do arquivo binário de máquina .hack. a ser escrito, passa essas informações para a classe Assemble. Essa classe que inicializa a tabela de símbolo (fillSymbolTable) e chama o método generateMachineCode para efetivamente gerar o arquivo de saída.
  - **Dependências** : Assemble.java
- Assemble
  - **Arquivo** : Assemble.java
  - **Descrição** : Classe responsável por criar o código de máquina, ela que **efetivamente** faz a varredura do arquivo .nasm de entrada e escreve o arquivo .hack de saída, gerando o código de máquina.
  - **Dependências** : Code.java, Parser.java, SymbolTable.java
- Code
  - **Arquivo** : Code.java
  - **Descrição** : Traduz mnemônicos da linguagem assembly para códigos binários da arquitetura Z0.
  - **Dependências** : none
- Parser
  - **Arquivo** : Parser.java
  - **Descrição** : 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.
  - **Dependências** : none
- SymbolTable
  - **Arquivo** : SymbolTable.java
  - **Descrição** : Mantém uma tabela com a correspondência entre os rótulos simbólicos e endereços numéricos de memória.
  - **Dependências** : none

Rubricas para avaliação de projetos

Cada integrante do grupo irá receber duas notas: uma referente ao desenvolvimento total do projeto (Projeto) e outra referente a sua participação individual no grupo (que depende do seu papel).

Projeto

Conceito	
I	- Menos da metade dos módulos funcionando
D	- Teste unitário ou Teste integração não passa
C	- Criado assembler a partir de estrutura de código disponibilizada - Todos os testes unitários passam no teste - Os testes de integração passam nos testes - Travis configurado corretamente
B	- Implementando modo verbose (-v) que possibilita analisar o assembler e suas etapas - Verifica se instrução de jump é seguida de NOP, caso contrário da erro
B+	- Insere automaticamente um NOP após intrução de JUMP que não é seguida de nop. Imprime mensagem de alerta que isso foi feito.
A	- Possui macros para facilitar desenvolvimento no assembly*
A+	- Código e documentação gerada com DoxyGen

- – Entregue atrasado = Conceito C

Macros

Suponha o seguinte arquivo .nasm :

```
%macro name nPar
... ..
%endmacro
```

- name : Nome do macro
- nPar : Quantidade de parâmetros (0, 1, 2), acessível via : par0, par1

O macro serve para facilitar o reaproveitamento de códigos, diferente de uma função que altera o fluxo de execução do código para o endereço que a função está salva, o macro funciona copiando o macro para a onde ele foi chamado, como no exemplo a seguir:

Código sem macro :

```
; carrega 5 em D
leaw $5, %A
movw %A, %D

; salva valor em RAM8
leaw $8, %A
movw %D, (%A)
```

Código com 2 macros definidos :

- movCntToReg : Move uma constante (par0) para uma registrador (par1)
- movRegtoRAM : Move um registrador (par0) para o endereço de memória (par1).

```
%macro movCntToReg 2
leaw par0, %A
movw %A, par1
%endmacro

%macro movRegtoRAM 2
  leaw par0, %A
  movw par1, (%A)
%endmacro

; carrega 5 em D
movCntToReg $D, %5
; chama macro para salvar em RAM 8
movRegtoRAM %D, $8
```

Desenvolvedor

Conceito	
Insatisfatório (I)	- Se comprometeu a fazer algum desenvolvimento (kanban) e não fez. - Criou o branch mas não fez o desenvolvimento completo.
Satisfatório (A)	- Desenvolveu as rotinas atribuídas pelo Facilitador para passarem nos testes!

Conceito	
	- Acompanhou o Kanban board (github project). Ex: Puxou tarefas, etc. - Submeteu alterações por pull requests.

Facilitador

Conceito	
Insatisfatório	
(I)	- Não acompanhou o projeto, deixando os colegas sozinhos. - O relatório das atividades não é condizente com o real desempenho dos integrantes (analisado via git).
Satisfatório	
(A)	- Atualizou o repositório pelo Fork. - Fez a manutenção do Kanban board (GitHub project). Ex: cria cards, atribui tarefas, da feedback de inssues. - Aceitou os pull-requests. - Resolveu conflitos de merge nos pull requests. - Acompanhou o desenvolvimento do grupo, dando o suporte sempre que necessário - **Entregou o branch master sem nenhum erro (passando no travis)** - Fez o relatório das atividades descrevendo o papel de cada integrante com clareza.