

PCS 2302/2024
 Laboratório de
 Fundamentos da
 Eng.de Computação

Professores:
 Anarosa A.F. Brandão
 Jaime S. Sichman
 Reginaldo Arakaki
 Ricardo L.A. Rocha
 © 2009

Aula 1:
 Introdução
 Máquina de Turing
 Máq. Von Neumann

Autores:
 Anna H. R. Costa
 Jaime S. Sichman
 João José Neto
 Paulo S. Muniz Silva
 Ricardo L. A. Rocha

Revisores:
 Diego Queiroz
 Tiago Matos

v. 1.4 ago 2009



PCS-2302 / PCS-2024
Lab. de Fundamentos de Eng. de Computação

Aula 01

Introdução
Máquina de Turing
Máquina de Von Neumann

Professores:
 Anarosa Alves Franco Brandão (PCS 2302)
 Jaime Simão Sichman (PCS 2302)
 Reginaldo Arakaki (PCS 2024)
 Ricardo Luís de Azevedo da Rocha (PCS 2024)

Monitores: Diego Queiroz e Tiago Matos

PCS 2302/2024
 Laboratório de
 Fundamentos da
 Eng.de Computação

Professores:
 Anarosa A.F. Brandão
 Jaime S. Sichman
 Reginaldo Arakaki
 Ricardo L.A. Rocha
 © 2009

Aula 1:
 Introdução
 Máquina de Turing
 Máq. Von Neumann

Autores:
 Anna H. R. Costa
 Jaime S. Sichman
 João José Neto
 Paulo S. Muniz Silva
 Ricardo L. A. Rocha

Revisores:
 Diego Queiroz
 Tiago Matos

v. 1.4 ago 2009

Roteiro

- Planejamento da disciplina
- Máquina de Turing
 - Introdução
 - Ações básicas de uma Máquina de Turing
 - Definição formal de uma Máquina de Turing
 - Linguagens decidíveis
- Aplicações da Máquina de Turing
 - Computação de funções
 - Um Simulador de Máquinas de Turing
 - Uma Máquina de Turing para Somar em Unário
- Máquina de Von Neumann
 - Problemas práticos da Máquina de Turing
 - Exemplo de uma máquina muito simples na arquitetura Von Neumann
 - Exemplo de um simulador de uma máquina de Von Neumann (MVN)
- Parte Experimental
 - Pequenos programas em código de máquina MVN



Planejamento da disciplina (1)

• Objetivos da disciplina

- Apresentar conceitos fundamentais da engenharia de computação, do ponto de vista do software, tendo os seguintes temas como motivação:
 - Máquina de Turing
 - Máquina de Von Neumann
 - Principais aspectos dos Programas de Sistema
- Criar e realizar modelos, abstrações e programas de sistema, empregando o paradigma da orientação a objetos



Planejamento da disciplina (2)



• Método

- Aulas ministradas em laboratório com:
 - Exposição conceitual dos problemas a resolver
 - Realização experimental dos conceitos apresentados para atender à meta da aula

• Componentes da Avaliação

- Média das notas dos produtos criados na aula (R)
- Média das notas de duas provas (P)

• Avaliação final = $(2P + R) / 3$

PCS 2302/2024
Laboratório de
Fundamentos da
Eng.de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha
© 2009

Aula 1:
Introdução
Máquina de Turing
Máq. Von Neumann

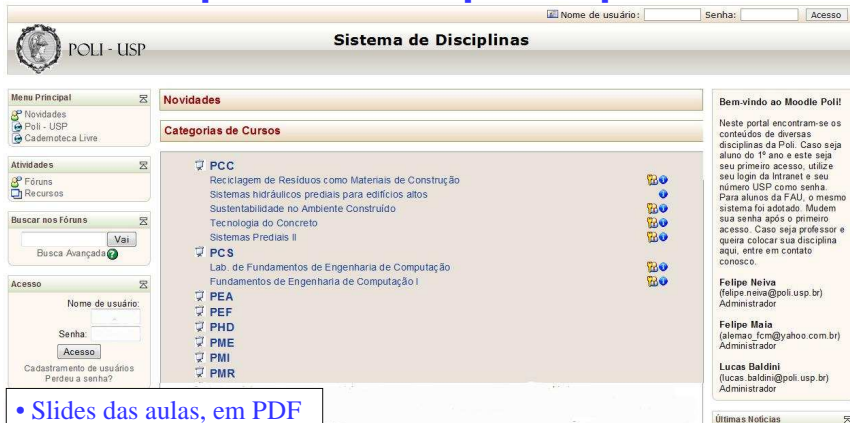
Autores:
Anna H. R. Costa
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Revisores:
Diego Queiroz
Tiago Matos



v. 1.4 ago 2009

Planejamento da disciplina (3)

Sítio : <http://www.edu.poli.usp.br/moodle/>



- Slides das aulas, em PDF
- Material didático de apoio
- Instruções e avisos
- Links úteis

PCS 2302/2024
Laboratório de
Fundamentos da
Eng.de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha
© 2009

Aula 1:
Introdução
Máquina de Turing
Máq. Von Neumann



Autores:
Anna H. R. Costa
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Revisores:
Diego Queiroz
Tiago Matos

v. 1.4 ago 2009

Planejamento da disciplina (4)

- Aula 1**
 - Máquinas de Turing (MT)
 - Exemplos de programas em um simulador da Máquina de Turing (MT)
 - Máquina de Von Neumann (MVN)
 - Exemplos de programas em um simulador da Máquina de Von Neumann (MVN)
 - Exercícios: Escrever e executar programas em um simulador da Máquina de Von Neumann (MVN)
- Aula 2**
 - Exercícios: Escrever e executar programas em um simulador da Máquina de Von Neumann (MVN)

PCS 2302/2024
 Laboratório de
 Fundamentos da
 Eng.de Computação

Professores:
 Anarosa A.F. Brandão
 Jaime S. Sichman
 Reginaldo Arakaki
 Ricardo L.A. Rocha
 © 2009

Aula 1:
 Introdução
 Máquina de Turing
 Máq. Von Neumann



Autores:
 Anna H. R. Costa
 Jaime S. Sichman
 João José Neto
 Paulo S. Muniz Silva
 Ricardo L. A. Rocha

Revisores:
 Diego Queiroz
 Tiago Matos

v. 1.4 ago 2009

Planejamento da disciplina (5)

- Aula 3**
 - Orientação a objetos: elementos básicos. Linguagem Java.
 - Exercícios: Estender exemplos de aula.
- Aula 4**
 - Orientação a objetos: elementos avançados.
 - Exercícios: Estender exemplos de aula.
- Aula 5**
 - Arquitetura básica do simulador MVN.
 - Exercícios: Implementação do simulador MVN (1).
- Aula 6**
 - Arquitetura estendida do simulador MVN.
 - Exercícios: Implementação do simulador MVN (2).

PCS 2302/2024
 Laboratório de
 Fundamentos da
 Eng.de Computação

Professores:
 Anarosa A.F. Brandão
 Jaime S. Sichman
 Reginaldo Arakaki
 Ricardo L.A. Rocha
 © 2009

Aula 1:
 Introdução
 Máquina de Turing
 Máq. Von Neumann

Autores:
 Anna H. R. Costa
 Jaime S. Sichman
 João José Neto
 Paulo S. Muniz Silva
 Ricardo L. A. Rocha

Revisores:
 Diego Queiroz
 Tiago Matos

v. 1.4 ago 2009

Planejamento da disciplina (6)

- Aula 7**
 - Visão geral de programação de sistemas:
Dumper e Loader
 - Exercícios: Implementar um *dumper* e *loader* para a MVN
- Aula 8**
 - 1a. Prova



Planejamento da disciplina (7)

- **Aula 9**
 - Linguagem simbólica. Especificação do montador absoluto para a MVN
 - Exercícios: Implementação do montador absoluto
- **Aula 10**
 - Especificação do montador relocável para a MVN
 - Exercícios: Implementação do montador relocável (1)
- **Aula 11**
 - Exercícios: Implementação do montador relocável (2)
- **Aula 12**
 - Especificação do ligador e relocador para a MVN
 - Exercícios: Implementar o ligador e o relocador
- **Aula 13**
 - 2a. Prova



Máquina de Turing (1)

- **Máquina de Turing**: modelo de computação proposto pelo inglês Alan M. Turing em 1936.



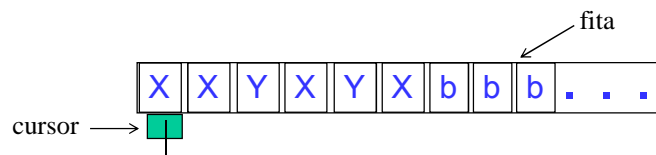
Alan M. Turing, disponível em
http://en.wikipedia.org/wiki/Alan_Turing



Máquina de Turing (2)

- Uma **Máquina de Turing** compõe-se de:

- Uma fita infinita, composta de células, cada qual contendo um símbolo de um alfabeto finito disponível (a fita também implementa a memória externa da máquina).
- Um cursor, que pode efetuar leitura ou escrita em uma célula, ou mover-se para a direita ou para a esquerda.
- Uma máquina de estados finitos, que controla o cursor.



**Máquina de Estados Finitos
(MEF)**



Computação em uma MT (3)

- Inicialmente a fita contém somente a cadeia de entrada, com o cursor posicionado (por convenção) no início da cadeia (o restante da fita está em branco “b”).
- Para armazenar algo, a máquina o grava na fita.
- Se a máquina tentar mover o cursor para a esquerda, estando o cursor posicionado na primeira célula da fita, este não se moverá.
- As saídas **aceita** e **rejeita** são obtidas ao entrar a máquina nos estados de aceitação e rejeição, respectivamente.
- Se a máquina não entrar em um estado de aceitação ou de rejeição, continuará sua computação para sempre (*loop* infinito).



MT como um Conjunto de Ações (4)

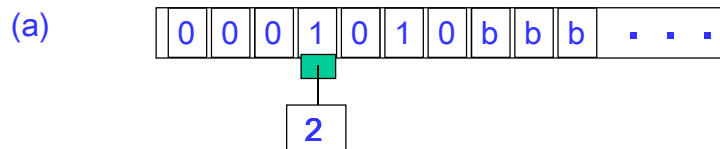
- Uma MT pode ser descrita por um conjunto de ações.
- **Ações: (s, i, i', s', d)** sendo:
 - s: estado corrente da MEF
 - i: símbolo que está sendo lido na fita
 - i': símbolo que é gravado na fita, no lugar de i
 - s': próximo estado da MEF
 - d ∈ {D,E}, indicando que o cursor pode se mover para a Direita ou para a Esquerda.



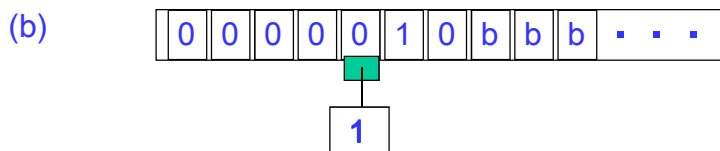
Ações da Máquina de Turing (1)

Exemplo:

Estando a máquina na situação (a):



executando a ação (2,1,0,1,D), a nova situação será (b):



(s, i, i', s', d)



Ações da Máquina de Turing (2)

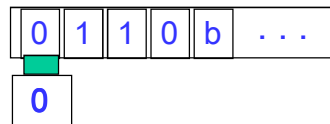
- Uma **Máquina de Turing** deve ser vista como um computador com um único programa fixo (MEF). Para alterar o programa, é preciso construir outra máquina.
- É possível construir uma **Máquina de Turing Universal**, a qual simula a computação de Máquinas de Turing arbitrárias sobre entradas arbitrárias.
- Eliminadas suas limitações de recursos, um **computador moderno** pode ser visto como um dispositivo similar à Máquina de Turing Universal.



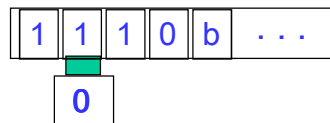
Ações da Máquina de Turing (3)

- Ex: Uma MT é definida pelo conjunto de quintuplas: $(0,0,1,0,D)$, $(0,1,0,0,D)$, $(0,b,1,1,E)$, $(1,0,0,1,D)$, $(1,1,0,1,D)$, $(1,b,b,2,D)$. O estado de aceitação é 2. Verificar sua computação:

Início:



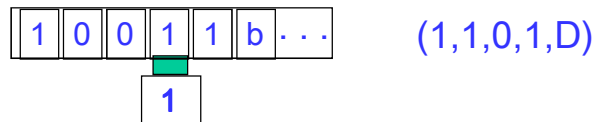
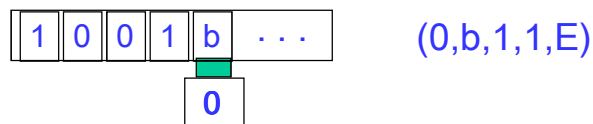
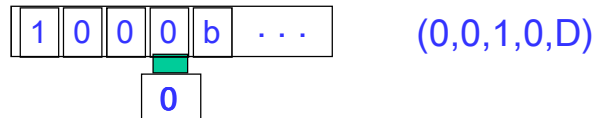
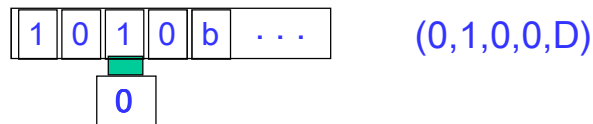
$(0,0,1,0,D)$



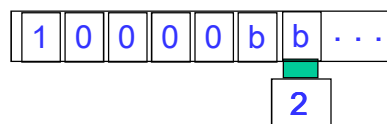
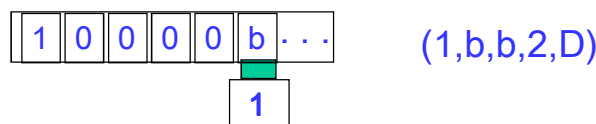
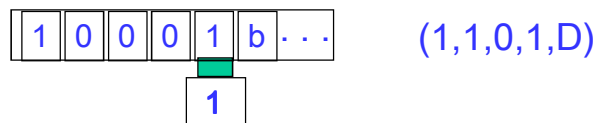
$(0,1,0,0,D)$



Ações da Máquina de Turing (4)



Ações da Máquina de Turing (5)



Quando a máquina parar no estado 2 (estado de aceitação), a fita conterá:





Definição Formal da Máquina de Turing

- Uma MT $(S, I, \Gamma, f, \sigma_0, \sigma_A, \sigma_R)$ compõe-se de:
 - Um conjunto finito S de estados;
 - Um conjunto finito I de símbolos de entrada, $b \notin I$;
 - Um conjunto finito Γ de símbolos da fita, com $b \in \Gamma$ e $I \subseteq \Gamma$;
 - Uma função $f: S \times \Gamma \rightarrow S \times \Gamma \times \{E, D\}$, sendo D : direita, E : esquerda;
 - $\sigma_0 \in S$ é o estado inicial;
 - $\sigma_A \in S$ é o estado de aceitação;
 - $\sigma_R \in S$ é o estado de rejeição, com $\sigma_A \neq \sigma_R$.
 - OBS: b : célula em branco da fita



Diagrama de Transições de uma MT (1)

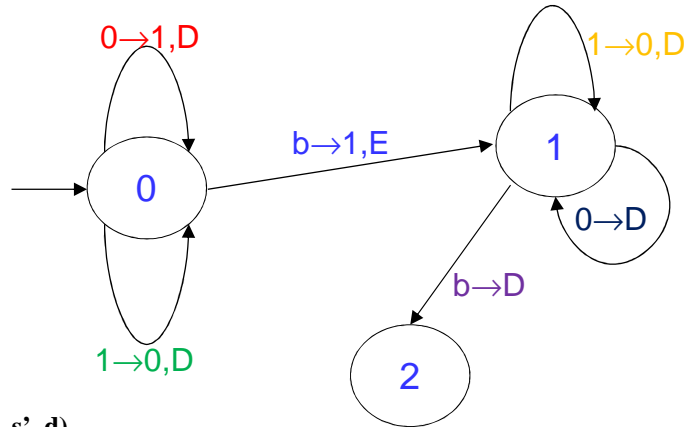
- Seja $M = (S, I, \Gamma, f, \sigma_0, \sigma_A, \sigma_R)$ uma **Máquina de Turing**. O **diagrama de transições** de M é um grafo orientado G com:
 - vértices membros de S .
 - uma seta indicando o **estado inicial** σ_0 .
 - uma aresta orientada (σ_1, σ_2) existe em G se existir uma entrada de fita $i \in \Gamma$ com $f(\sigma_1, i) = (\sigma_2, j, m)$, sendo $\sigma_1, \sigma_2 \in S$, $j \in \Gamma$ e m indica o movimento do cursor, $m \in \{E, D\}$.

Neste caso, a aresta (σ_1, σ_2) é rotulada com $i \rightarrow j, m$ para $i \neq j$, e $i \rightarrow m$ para $i = j$.



Diagrama de Transições de uma MT (2)

Desenhar o diagrama de transições da MT definida pelo conjunto de ações seguinte: $\{(0,0,1,0,D), (0,1,0,0,D), (0,b,1,1,E), (1,0,0,1,D), (1,1,0,1,D), (1,b,b,2,D)\}$.

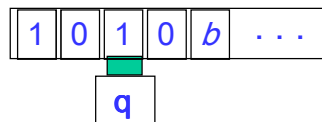


(s, i, i', s', d)



Configuração de uma MT (1)

- Conforme a MT efetua sua computação, mudanças ocorrem no estado atual, no conteúdo da fita e na posição do cursor.
- Uma situação da computação pode ser representada por uma **configuração**, que descreve de forma abreviada a situação da máquina.
- Por exemplo: a configuração **10q10** representa a seguinte situação:





Configuração de uma MT (2)

- A configuração inicial de uma MT é dada por $\sigma_0 w$, para a cadeia de entrada w .
- O estado, numa configuração de aceitação, deve ser σ_A .
- O estado, numa configuração de rejeição, deve ser σ_R .
- Configurações de aceitação e de rejeição constituem sempre **configurações de parada** da MT.



Aceitação de Cadeias na MT

- Uma MT **aceita** a entrada w se existir alguma sequência de configurações C_1, C_2, \dots, C_k tal que:
 - C_1 é a configuração inicial da MT, $\sigma_0 w$;
 - Aplicando f a C_i , obtém-se C_{i+1} ;
 - C_k é uma configuração de aceitação.
- A coleção de cadeias que a Máquina de Turing M aceita é a **linguagem** definida pela máquina M , e é denotada por $L(M)$.



Linguagem Decidível

- Ao iniciar uma MT com uma entrada, pode-se aceitar a entrada ou não aceitar a entrada.
- Uma MT pode **não aceitar** uma entrada:
 1. ao entrar em σ_R e **rejeitar** a cadeia ou
 2. ao entrar em *loop* infinito e **não parar nunca**.
Pode ser muito difícil distinguir uma MT que entrou em *loop* infinito de outra que está demorando para completar a computação.
- Uma MT **decide uma linguagem** quando pára para todas as suas possíveis entradas:
 - se $w \notin L(M)$, MT pára em σ_R ;
 - se $w \in L(M)$, MT pára em σ_A .



Exemplo de MT (1)

Projeto de uma Máquina de Turing M que decide a linguagem $A = \{0^n 1^n \mid n > 0\}$:

Algoritmo esquematizado de M:

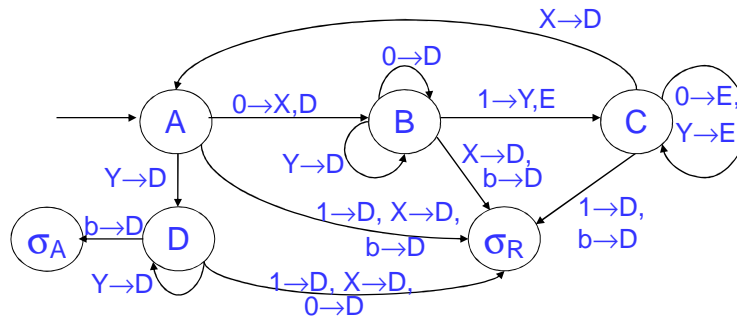
$M =$ "para a cadeia de entrada $w \in \{0,1\}^*$:

1. Se $w = \lambda$, rejeite.
2. Leia a célula corrente
Se o conteúdo for 0, troque por X e vá para o passo 3.
Se o conteúdo for Y, procure o final da cadeia de Ys e aceite.
Se não, rejeite.
1. Vá para a direita da fita, procurando o primeiro 1.
Se encontrar, troque-o por Y e vá para o passo 4.
Se não, rejeite.
1. Vá para a esquerda na fita, procurando por um X.
Se encontrar, volte a avançar na fita e vá para o passo 2.
Se não, rejeite."



Exemplo de MT (2)

- Descrição formal de $M = (S, I, \Gamma, f, \sigma_0, \sigma_A, \sigma_R)$:
 $S = \{A, B, C, D, \sigma_A, \sigma_R\}$, $\sigma_0 = A$, $I = \{0, 1\}$, $\Gamma = \{0, 1, X, Y, b\}$,
 $f =$ diagrama de transições abaixo:



Variantes da MT

- Um autômato finito é um caso muito particular de MT, que sempre grava o símbolo lido na célula, sempre se move para a direita e sempre pára ao ler o símbolo b (final da cadeia).
- Existem inúmeros modelos variantes de MT, porém todos se mostram equivalentes em termos de poder de computação (prova-se).
- Muitos outros modelos de computação de propósito geral também foram propostos, alguns bem diferentes de MT. No entanto, é possível provar que todos podem simular uns aos outros e, portanto, também são equivalentes em termos de poder computacional!
- ➔ Implicação importante: a classe de algoritmos que todos descrevem é única!



Aplicações da Máquina de Turing

- Vimos que uma MT pode decidir uma linguagem. No entanto, uma MT também pode **computar funções**:

Dada uma MT M e uma cadeia α , começamos com M na configuração inicial padrão em uma fita contendo α . Se M em algum momento pára deixando uma cadeia β na fita, podemos definir β como sendo o valor de uma função avaliada em α . Assim, $M(\alpha) = \beta$. Nesse caso, o **domínio da função** M compreende todas as cadeias α para as quais M , em algum momento, pára.



Algoritmos e MTs

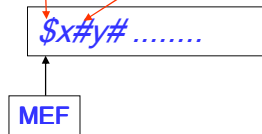
- Noção intuitiva de algoritmo:** é um conjunto finito de instruções que podem ser executadas mecanicamente em tempo finito para resolver algum problema. Com dados de entrada apropriados ao problema, o algoritmo deve obrigatoriamente parar e produzir como saída a resposta correta.
- Assim, qualquer função f , computável por uma MT, será uma função cujos valores podem ser determinados pela execução de um algoritmo ou procedimento computacional.



MT para Computar Funções (1)

- Exemplo 1: Projeto de uma MT que calcula a soma $x + y$. Considere x e y em representação unária, fornecidos na fita da seguinte forma:

Início da fita *Separador*



unária	decimal
1	0
11	1
111	2
1111	3
.....

Exemplo: $x=3, y=2$

fita: \$1111#111# ...

(Formato 1.1) Resposta na fita: \$111111#...

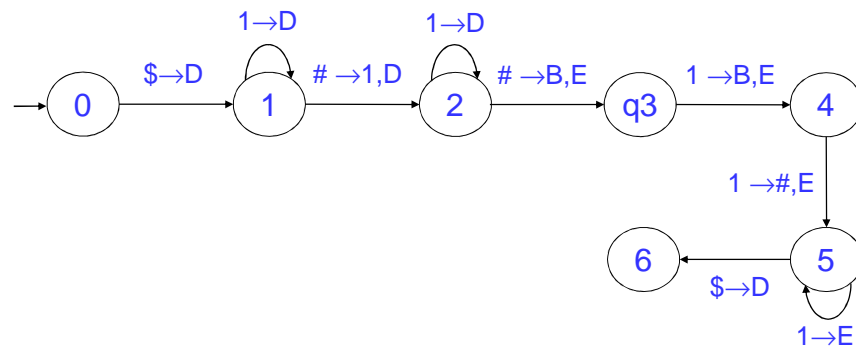
(Formato 1.2) Resposta na fita: \$1111#111#111111# ...



MT para Computar Funções (2)

- Uma solução possível (formato 1.1) é:

Diagrama de transições:





MT para Computar Funções (3)

• Realização

ATM

PCS 2302/2024 X+Y, soma 2 números unários (formato 1.1)

```
1 # $           // alfabeto de entrada: $ é o início da fita, # é o separador
1 B # $         // alfabeto da fita: B é branco (default)
1              // número de fitas
1              // número de trilhas na fita 0
2              // fita 0 é infinita nas duas direções
q0             // estado inicial
q6             // estado final
q0 $ q1 $ R    // q0 - início da fita, move para a direita
q1 1 q1 1 R    // q1 - se X tiver um dígito unário válido, move para a direita
q1 # q2 1 R    // q1 - final de X, escreve 1 e move para a direita
q2 1 q2 1 R    // q2 - se Y tiver um dígito unário válido, move para a direita
q2 # q3 B L    // q2 - final de Y, escreve B e move para a esquerda
q3 1 q4 B L    // q3 - último dígito de Y, escreve B e move para a esquerda
q4 1 q5 # L    // q4 - penúltimo dígito de Y, escreve # e move para a esquerda
q5 1 q5 1 L    // q5 - move para esquerda até o início da fita
q5 $ q6 $ R    // q5 - início da fita, move para a direita e pára
end            // final da máquina
```

Prólogo

Ações



Problemas Práticos da Máquina de Turing

- A **Máquina de Turing** se apresenta através de um formalismo poderoso, com fita infinita e apenas quatro operações triviais: ler, gravar, avançar e recuar.
- Isso faz dela um dispositivo detalhista que oferece apenas uma **visão microscópica** da solução do problema que pretende resolver, não permitindo ao usuário usar abstrações.
- Embora a Máquina de Turing Universal permita uma espécie de programação, o seu código é extenso e a sua velocidade final de execução, muito baixa.



A idéia da Máquina de Von Neumann (1)

- O **Modelo de Von Neumann** procura oferecer uma alternativa prática, disponibilizando ações mais poderosas e ágeis em seu repertório de operações.
- Isso viabiliza, para os mesmos programas, codificações muito mais expressivas, compactas e eficientes.



A idéia da Máquina de Von Neumann (2)

- Para isso, a Máquina de Von Neumann utiliza:
 - **Memória endereçável**, usando acesso aleatório
 - **Programa armazenado** na memória, para definir diretamente a função corrente da máquina (ao invés da MEF)
 - **Dados** representados na memória (ao invés da fita)
 - Codificação numérica **binária** em lugar da unária
 - **Instruções variadas e expressivas** para a realização de operações básicas muito freqüentes (ao invés de sub-máquinas específicas)
 - **Maior flexibilidade** para o usuário, permitindo operações de entrada e saída, comunicação física com o mundo real e controle dos modos de operação da máquina



Elementos da Arquitetura a Simular (1)

- Neste curso pretende-se simular um *processador muito simples*, porém estruturalmente similar aos disponíveis na realidade
- O processador tem um conjunto de elementos físicos de armazenamento de informações
 - Memória Principal:** para armazenar programas e dados
 - Acumulador (AC):** funciona como área de trabalho, para a execução de operações aritméticas e lógicas
 - Outros **registradores auxiliares:** empregados em diversas operações intermediárias no processamento dos programas
- O conjunto de dados neles contidos em cada instante constitui o **estado instantâneo** do processamento

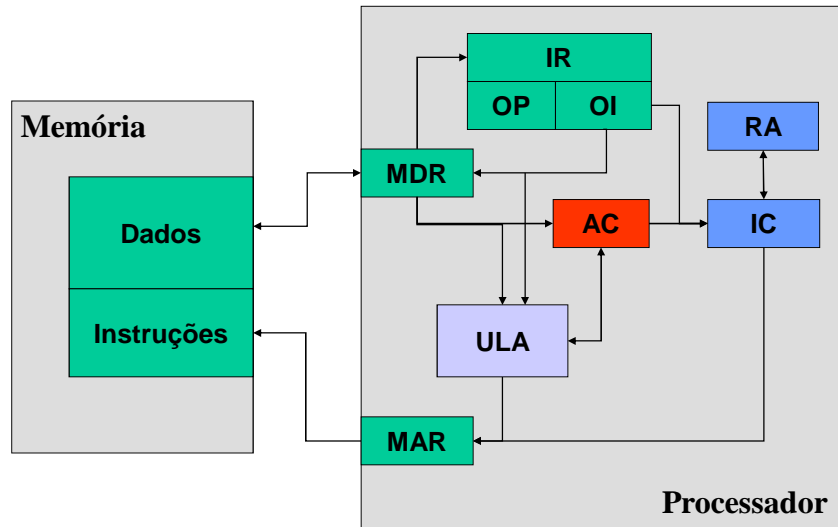


Elementos da Arquitetura a Simular (2)

- Os **Registradores Auxiliares** são:
 - Registrador de Dados da Memória (MDR)** – serve como ponte para os dados que trafegam entre a memória e os outros elementos da máquina
 - Registrador de Endereço da Memória (MAR)** – indica qual é a origem ou o destino, na memória principal, dos dados contidos no registrador de dados da memória.
 - Registrador de Endereço da Próxima Instrução (IC)** – indica em cada instante qual será a próxima instrução a ser executada pelo processador.
 - Registrador de Instrução** – contém a instrução em execução
 - Código de Operação** – parte do registrador de instrução que identifica a instrução que está sendo executada
 - Operando da Instrução** – complementa a instrução indicando o dado ou o endereço sobre o qual ela deve agir.
 - Registrador de Endereço de Retorno** – guarda o endereço de retorno da sub-rotina ou função em execução.

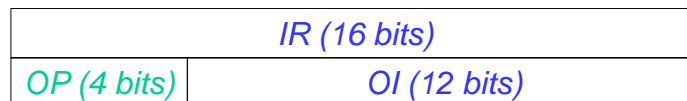


Diagrama da Arquitetura a Simular (3)



Conjunto de registradores da Máquina de Von Neumann (MVN)

MAR	Registrador de endereço de memória
MDR	Registrador de dados da memória
IC	Registrador de endereço da próxima instrução
IR	Registrador de instrução
OP	Registrador de código de operação
OI	Registrador de operando de instrução
RA	Registrador de endereço de retorno
AC	Acumulador





Funcionamento de um Simulador

Devem-se separar dois conceitos independentes na lógica de um simulador:

- **Comandos de controle do simulador:** esta parte independe da arquitetura do computador que se está simulando, e sua função é de orientar a operação do programa simulador e de permitir ao usuário observar e alterar o conteúdo dos componentes do processador simulado.
- **Execução das instruções do processador simulado:** esta parte do simulador depende fortemente da arquitetura da máquina cuja operação se deseja simular, que deve implementar um modelo da máquina simulada, no nível de granularidade mais conveniente em cada caso.



Comandos de Controle do Simulador

- Conta-se com os seguintes comandos de controle para o programa simulador:
 - **[INITIALIZE]** – atribui valores iniciais padrão a todos os elementos importantes do simulador e da arquitetura.
 - **[LOAD]** – serve para carregar programas e dados para a memória da máquina simulada
 - **[STEP]** – serve colocar o simulador no modo de operação passo a passo.
 - **[RUN]** – serve colocar o simulador no modo de operação contínuo.
 - **[EXECUTE]** – serve para promover a execução do programa, conforme o modo de operação: execução contínua/uma instrução por vez.
 - **[SHOW]** – serve para mostrar o conteúdo das memórias da máquina simulada, após a execução de um passo (modo STEP) ou após a execução de um programa (modo RUN).



[EXECUTE] – Obtenção e Decodificação

EXECUTE - Serve para promover a execução do programa, conforme o modo de operação: contínua/ou uma instrução por vez

1) Determinação da Próxima Instrução a Executar

2) Fase de Obtenção da Instrução

•Obter na memória, no endereço contido no registrador de Endereço da Próxima Instrução, o código da instrução desejada

3) Fase de Decodificação da Instrução

•Decompor a instrução em duas partes: o código da instrução e o seu operando, depositando essas partes nos registradores de instrução e de operando, respectivamente.

•Selecionar, com base no conteúdo do registrador de instrução, um procedimento de execução dentre os disponíveis no repertório do simulador (passo 4).



Conjunto de instruções da Máquina de Von Neumann (MVN)

Código (hexa)	Instrução	Operando
0	Desvio incondicional	endereço do desvio
1	Desvio se acumulador é zero	endereço do desvio
2	Desvio se acumulador é negativo	endereço do desvio
3	Deposita uma constante no acumulador	constante relativa de 12 bits
4	Soma	endereço da parcela
5	Subtração	endereço do subtraendo
6	Multipliação	endereço do multiplicador
7	Divisão	endereço do divisor
8	Memória para acumulador	endereço-origem do dado
9	Acumulador para memória	endereço-destino do dado
A	Desvio para subprograma (função)	endereço do subprograma
B	Retorno de subprograma (função)	endereço do resultado
C	Parada	endereço do desvio
D	Entrada	dispositivo de e/s (*)
E	Saída	dispositivo de e/s (*)
F	Chamada de supervisor	constante (**)

(*) ver slide seguinte

(**) por ora, este operando (tipo da chamada) é irrelevante, e esta instrução nada faz.



[EXECUTE] – Execução de instrução (1)

4) Fase de Execução da Instrução

- Executar o procedimento selecionado em 3, usando como operando o conteúdo do registrador de operando, preenchido anteriormente.
- Caso a instrução executada não seja de desvio, incrementar o registrador de endereço da próxima instrução a executar. Caso contrário, o procedimento de execução já terá atualizado convenientemente tal informação.

4.1) Execução da instrução (decodificada em 3)

- De acordo com o código da instrução a executar (contido no registrador de instrução), executar os procedimentos de simulação correspondentes (detalhados adiante)

4.2) Acerto do registrador de Endereço da Próxima Instrução para apontar a próxima instrução a ser simulada:

- Incrementar o registrador de Endereço da Próxima Instrução.



[EXECUTE] – Execução de instrução (2)

- Obs.: Sistema de **numeração e aritmética** adotada: Binário, em complemento de dois
 - representa inteiros e executa operações em 16 bits.
 - o bit mais à esquerda é o bit de sinal (1 = negativo)

Registrador de instrução = 0 (desvio incondicional)

- modifica o conteúdo do registrador de Endereço da Próxima Instrução (**IC**) armazenando nele o conteúdo do registrador de operando (**OI**)

$IC := OI$

Registrador de instrução = 1 (desvio se acumulador é zero)

- se o conteúdo do acumulador for zero, então modifica o conteúdo do registrador de Endereço da Próxima Instrução (**IC**), armazenando nele o conteúdo do registrador de operando (**OI**)

Se $AC = 0$ então $IC := OI$ se não $IC := IC + 1$



[EXECUTE] – Execução de instrução (3)

Registrador de instrução = 2 (desvio se negativo)

- se o conteúdo do acumulador (**AC**) for negativo, isto é, se o bit mais significativo for 1, então modifica o conteúdo do registrador de Endereço da Próxima Instrução (**IC**) armazenando nele o conteúdo do registrador de operando (**OI**)

Se $AC < 0$ então $IC := OI$ se não $IC := IC + 1$

Registrador de instrução = 3 (constante para acumulador)

- Armazena no acumulador (**AC**) o número relativo de 12 bits contido no registrador de operando (**OI**), estendendo seu bit mais significativo (bit de sinal) para completar os 16 bits do acumulador

$AC := OI$

$IC := IC + 1$



[EXECUTE] – Execução de instrução (4)

Registrador de instrução = 4 (soma)

- Soma ao conteúdo do acumulador (**AC**) o conteúdo da posição de memória indicada pelo registrador de operando **MEM[OI]**
- Guarda o resultado no acumulador

$AC := AC + MEM[OI]$

$IC := IC + 1$

Registrador de instrução = 5 (subtração)

- Subtrai do conteúdo do acumulador (**AC**) o conteúdo da posição de memória indicada pelo registrador de operando **MEM[OI]**

- Guarda o resultado no acumulador

$AC := AC - MEM[OI]$

$IC := IC + 1$



[EXECUTE] – Execução de instrução (5)

Registrador de instrução = 6 (multiplicação)

- Multiplica o conteúdo do acumulador (**AC**) pelo conteúdo da posição de memória indicada pelo registrador de operando **MEM[OI]**
- Guarda o resultado no acumulador

$$AC := AC * MEM[OI]$$

$$IC := IC + 1$$

Registrador de instrução = 7 (divisão inteira)

- Dividir o conteúdo do acumulador (**AC**) pelo conteúdo da posição de memória indicada pelo registrador de operando **MEM[OI]**
- Guarda a parte inteira do resultado no acumulador

$$AC := \text{int} (AC / MEM[OI])$$

$$IC := IC + 1$$


[EXECUTE] – Execução de instrução (6)

Registrador de instrução = 8 (memória para acumulador)

- Armazena no acumulador (**AC**) o conteúdo da posição de memória cujo endereço é o conteúdo do registrador de operando **MEM[OI]**

$$AC := MEM[OI]$$

$$IC := IC + 1$$

Registrador de instrução = 9 (acumulador para memória)

- Guarda o conteúdo do acumulador (**AC**) na posição de memória indicada pelo registrador de operando **MEM[OI]**

$$MEM[OI] := AC$$

$$IC := IC + 1$$



[EXECUTE] – Execução de instrução (7)

Registrador de instrução = A (desvio para subprograma)

- Armazena o conteúdo do registrador de Endereço da Próxima Instrução (**IC**), incrementado de uma unidade, no endereço do subprograma
- Armazena no registrador de Endereço da Próxima Instrução (**IC**) o conteúdo do registrador de operando incrementado de uma unidade (**OI**)

$RA := IC$

$IC := OI$

Registrador de instrução = B (retorno de subprograma)

- Armazena no registrador de Endereço da Próxima Instrução (**IC**) o conteúdo do registrador de endereço de retorno (**RA**), e no registrador acumulador (**AC**) o conteúdo da posição de memória apontada pelo registrador de operando $MEM[OI]$

$AC := MEM[OI]$

$IC := RA$



[EXECUTE] – Execução de instrução (8)

Registrador de instrução = C (stop)

- Modifica o conteúdo do registrador de Endereço da Próxima Instrução (**IC**) armazenando nele o conteúdo do registrador de operando (**OI**)

$IC := OI$

Registrador de instrução = D (input)

- Aciona o dispositivo padrão de entrada e aguardar que o usuário forneça o próximo dado a ser lido
- Transfere o dado para o acumulador

aguarda

$AC := \text{dado de entrada}$

$IC := IC + 1$



[EXECUTE] – Execução de instrução (9)

Registrador de instrução = E (output)

- Transfere o conteúdo do acumulador (AC) para o dispositivo padrão de saída.
- Aciona o dispositivo padrão de saída e aguardar que este termine de executar a operação de saída

dado de saída := AC

aguarda

IC := IC + 1

Registrador de instrução = F (supervisor call)

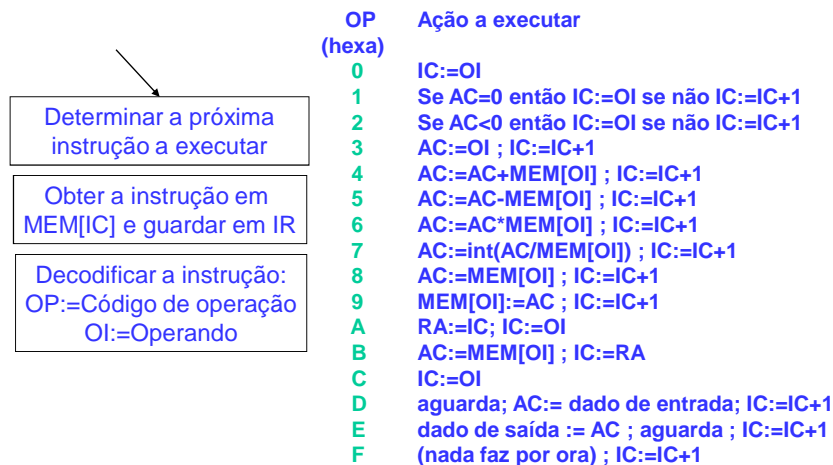
(não implementado: por enquanto esta instrução não faz nada)



IC := IC + 1



Diagrama de fluxo do Interpretador [detalhamento de EXECUTA]

Executa *uma* instrução



PCS 2302/2024
Laboratório de
Fundamentos da
Eng.de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha
© 2009

Aula 1:
Introdução
Máquina de Turing
Máq. Von Neumann

Autores:
Anna H. R. Costa
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Revisores:
Diego Queiroz
Tiago Matos

v. 1.4 ago 2009



Conjunto de registradores da Máquina de Von Neumann (MVN)

Operações de Entrada e Saída

<i>OP</i>	<i>Tipo</i>	<i>Dispositivo</i>
OP Tipo	D (entrada) ou E (saída) Tipos de dispositivo: 0 = Teclado 1 = Monitor 2 = Impressora 3 = Disco	
Dispositivo	Identificação do dispositivo. Pode-se ter vários tipos de dispositivo, ou unidades lógicas (LU). No caso do disco, um arquivo é considerado uma unidade lógica.	

Pode-se ter, portanto, até 16 tipos de dispositivos e, cada um, pode ter até 256 unidades lógicas.

57

PCS 2302/2024
Laboratório de
Fundamentos da
Eng.de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha
© 2009

Aula 1:
Introdução
Máquina de Turing
Máq. Von Neumann

Autores:
Anna H. R. Costa
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Revisores:
Diego Queiroz
Tiago Matos

v. 1.4 ago 2009

Exemplo de Programa – Prog1

- Problema: Somar o valor de duas variáveis iniciadas com os valores -125_{10} e 100_{10} , colocando o resultado em outra variável.

```

; prog1.mvn
; Soma os valores de duas posições de memória e guarda o
; resultado em outra posição de memória, parando na
; instrução final.
;
00 0008 ; Ponto de entrada: JMP para as instruções
; Variáveis locais
02 FF83 ; A = 0xFF83 (-125)
04 0064 ; B = 0x0064 (100)
06 0000 ; RESULTADO deverá ser 0xFFE7 (-25)
; Instruções do programa
08 8002 ; Carrega o conteúdo de A no acumulador
0A 4004 ; Adiciona B ao conteúdo do acumulador
0C 9006 ; Armazena o resultado em RESULTADO
0E C00E ; Pára em 0x000E
          
```

58



Execução de Programa – Prog1

```
C:\Users\humberto.sandmann\Workspace\mun\classes>java mun.MonPcs
MUN Inicializada

Escola Politecnica da Universidade de Sao Paulo
PCS2302/PCS2024 Simulador da Maquina de Von Neumann
MUN versao 3.5a (Julho/2008) - Todos os direitos reservados

COMANDO SINTAXE          OPERACAO
-----
h                          Ajuda
b                          Ativa/Desativa modo Debug
i                          Inicializa MUN
v                          Obtem versao da MUN
s                          Manipula dispositivos de I/O
g                          Lista conteudo dos registradores
p [arq]                   Carrega programa ASCII para memoria
m [ini] [fin] [arq]       Lista conteudo da memoria
l                          Loader
d                          Dumper
r [addr] [regs]           Executa programa em um determinado endereco
x                          Finaliza MUN e terminal

> p
Informe o nome do arquivo de entrada: ../prog1.mvn
programa ../prog1.mvn carregado

> r
Endereco atual do IC: 0000. Informe (novo) endereco do IC:
Exibir valores dos registradores a cada passo do ciclo FDE (s/n)[s]:
IC  IR  AC  MAR  MBR  OP  RA  OI  IOI1
=====
0008 0008 0000 0000 0008 0000 0000 0008 8002
000a 8002 ff83 0008 8002 0008 0000 0002 ff83
000c 4004 ffe7 000a 4004 0004 0000 0004 0064
000e 9006 ffe7 000c 9006 0009 0000 0006 ffe7
000e c00e ffe7 000e c00e 000c 0000 000e c00e

>
```



Exemplo de Programa – Prog2 (1)

- Problema: Implementar uma sub-rotina que subtrai dois inteiros. Os valores dos argumentos estão armazenados em duas variáveis do programa principal. O resultado é armazenado em uma variável do programa principal.

```
; prog2.mvn
; Programa de ilustração para JSR
; int subtrair(int x, int y) {
;     return x - y;
; }
;
00 0010 ; JMP início das instruções
; Variáveis locais
02 0010 ; A = 0x0010
04 0064 ; B = 0x0064
06 0000 ; RESULTADO Resultado de subtrair()
```



Exemplo de Programa – Prog2 (2)

```
; Programa principal
; Chamando SUBTRAIR(A, B)
;
10 8002 ; Carrega o conteúdo de A no acumulador
12 903C ; Armazena no parâmetro X
14 8004 ; Carrega o conteúdo de B
16 903E ; Armazena no parâmetro Y
18 A040 ; Chama a sub-rotina SUBTRAIR
1A 9006 ; Armazena o resultado em RESULTADO
1C C01C ; Pára em 0x001C
; Sub-rotina SUBTRAIR
; Parâmetros formais
3C 0000 ; X
3E 0000 ; Y
40 0000 ; Endereço de retorno
42 803C ; Carrega o conteúdo de X
44 503E ; Subtrai Y
46 B040 ; Retorna para o endereço posto em 0x0040
```



Execução de Programa – Prog2

```
> p ../prog2.mvn
programa ../prog2.mvn carregado
>
> r
Endereço atual do IC: 000e. Informe <novo> endereço do IC: 0000
Exibir valores dos registradores a cada passo do ciclo FDE <s/n>[s]:
IC IR AC MAR MBR OP RA OI IOI]
=====
0010 0010 0000 0000 0010 0000 0000 0010 0002
0012 0002 0010 0010 0002 0008 0000 0002 0010
0014 903c 0010 0012 903c 0009 0000 003c 0010
0016 8004 0064 0014 8004 0008 0000 0004 0064
0018 903e 0064 0016 903e 0009 0000 003e 0064
0042 a040 0064 0018 a040 000a 0000 0040 001a
0044 803c 0010 0042 803c 0008 0000 003c 0010
0046 503e ffac 0044 503e 0005 0000 003e 0064
001a b040 ffac 0046 b040 000b 0000 0040 001a
001c 9006 ffac 001a 9006 0009 0000 0006 ffac
001c c01c ffac 001c c01c 000c 0000 001c c01c
>
```



Algumas práticas de programação (1)

- O conjunto de instruções desta máquina de Von Neumann é extremamente limitado, exigindo alguns artifícios para a obtenção dos efeitos necessários:
 - Não há operações lógicas. Tudo deve ser feito com operações aritméticas.
 - Não há endereçamento indireto nem indexado. Tudo deve ser feito alterando-se convenientemente as instruções disponíveis, no próprio programa, antes de executá-las.
 - Incrementos e decrementos de variáveis devem ser feitos somando-se ou subtraindo-se as constantes desejadas (tipicamente 1 ou 2) às variáveis-alvo.



Algumas práticas de programação (2)

- Não há instruções específicas para todos os testes. Tudo deve ser feito combinando-se as instruções de desvios condicionais e usando-se lógica invertida quando necessário.
- Convém separar sub-rotinas já testadas e muito usadas, bem como variáveis e constantes, dos programas em desenvolvimento.





Algumas práticas de programação (3)

- Esta MVN suporta endereçamento de 12bits
- À medida que os programas ficam maiores e/ou tem-se mais de um programa na memória, é importante planejar um **mapa de memória**. Uma sugestão para um mapa bem simples é reservar os endereços 0x0000 – 0x01FF para área de dados, constantes, tabelas, etc., e os endereços a partir de 0x0200 para programas principais e sub-rotinas.
- Projete sempre no papel seus programas em linguagem de máquina, e simule seu funcionamento no papel antes de utilizar o computador. Economiza-se muito tempo e esforço evitando-se a depuração de erros na base da tentativa e de testes.



Algumas práticas de programação (4)

- Documente todos os programas desenvolvidos com comentários informativos no código, e no papel, com diagramas de fluxo e com desenhos ilustrativos das estruturas de dados utilizadas e das operações efetuadas. Em programação binária, é muito raro que, passados alguns dias, mesmo o autor consiga lembrar-se exatamente de como funciona o programa que ele próprio criou.
- Projete bem e anote os testes realizados e os resultados esperados. É freqüente ter de repeti-los para as novas versões de um programa em desenvolvimento.

PCS 2302/2024
Laboratório de
Fundamentos de
Eng.de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha
© 2009

Aula 1:
Introdução
Máquina de Turing
Máq. Von Neumann

Autores:
Anna H. R. Costa
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Revisores:
Diego Queiroz
Tiago Matos

v. 1.4 ago 2009

Exercícios

Comente devidamente os programas desenvolvidos.

GYXXA01E01_09 - Execute os programas 2302_aula01_prog1.mvn e 2302_aula02_prog2.mvn. Faça testes para valores diferentes de entradas. Guarde o mapa de memória antes e depois da execução.

GYXXA01E02_09 - Desenvolva uma sub-rotina (e um programa principal para testá-la) 2302_aula01_prog3.mvn para calcular uma potência inteira $k \geq 0$ de um número inteiro dado n , ou seja, n^k . Ensaie para alguns valores de n e de k (variáveis do programa principal), e verifique os resultados. (sala)

GYXXA01E03_09 - Construa um programa 2302_aula01_prog4.mvn que determina se um triângulo com lados a , b , c inteiros positivos é acutângulo, retângulo ou obtusângulo (usar o teorema de Pitágoras). (casa)

67




PCS 2302/2024
Laboratório de
Fundamentos de
Eng.de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha
© 2009

Aula 1:
Introdução
Máquina de Turing
Máq. Von Neumann

Autores:
Anna H. R. Costa
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Revisores:
Diego Queiroz
Tiago Matos

v. 1.4 ago 2009

Bibliografia (Programação de Sistemas)

Relíquias Preciosas

Barron, D. W. *Assemblers and Loaders* (3rd. ed.) MacDonald/Elsevier, 1978

Beck, L. L. *System Software - An Introduction to Systems Programming* Addison-Wesley, 1985

Calingaert, P. *Assemblers, Compilers and Program Translation* Computer Science Press, 1979

Donovan, J. J. *Systems Programming* McGraw-Hill, 1972

Duncan, F.G. *Microprocessor Programming and Software Development* Prentice Hall, 1979.

Freeman, P. *Software System Principles* SRA, 1975

Gear, C. W. *Computer Organization and Programming (3rd. ed.)* McGraw-Hill, 1981

Graham, R. M. *Principles of Systems Programming* John Wiley & Sons, 1975

Gust, P. *Introduction to Machine and Assembly Language Programming* Prentice Hall, 1986

Maginnis, J. B. *Elements of Compiler Construction* Appleton-Century-Crofts, Meredith Co., 1972

Presser, L. and White, J. R. *Linkers and Loaders* ACM Comp. Surveys, vol. 4, n. 3, pp. 149-168

Rosen, S. (ed.) *Programming Systems and Languages* McGraw-Hill, 1967

Tseng, V. (ed.) *Microprocessor Development and Development Systems* McGraw-Hill, 1982

Ullman, J. D. *Fundamental Concepts of Programming Systems* Addison-Wesley, 1976

Wegner, P. *Progr. Languages, Inf. Structures and Machine Organization* McGraw-Hill, 1968.

Welsh, J. and McKeag, M. *Structured System Programming* Prentice-Hall, 1980

68



Referências Bibliográficas

Costa, A.H.R., Sato, L.M., Sichman, J.S., Tori, R. *Material didático da disciplina PCS 2214 – Fundamentos da Engenharia de Computação I*, PCS/EPUSP, São Paulo, SP. 2004-2005.

Sipser, M. *Introduction to the Theory of Computation*. PWS Publishing Company, Boston, MA. 1997.

Leitura complementar:

**UM SIMULADOR-INTERPRETADOR PARA A
LINGUAGEM DE MÁQUINA DO PATINHO FEIO.**

**(João José Neto, Aspectos do Projeto de
Software de um Minicomputador, Dissertação de
Mestrado, EPUSP, S. Paulo, 1975, cap.3)**