lasmas

IAS Machine Simulator

Explicação das partes principais do simulador

COMO FUNCIONA?

Divisão da arquitetura IAS original

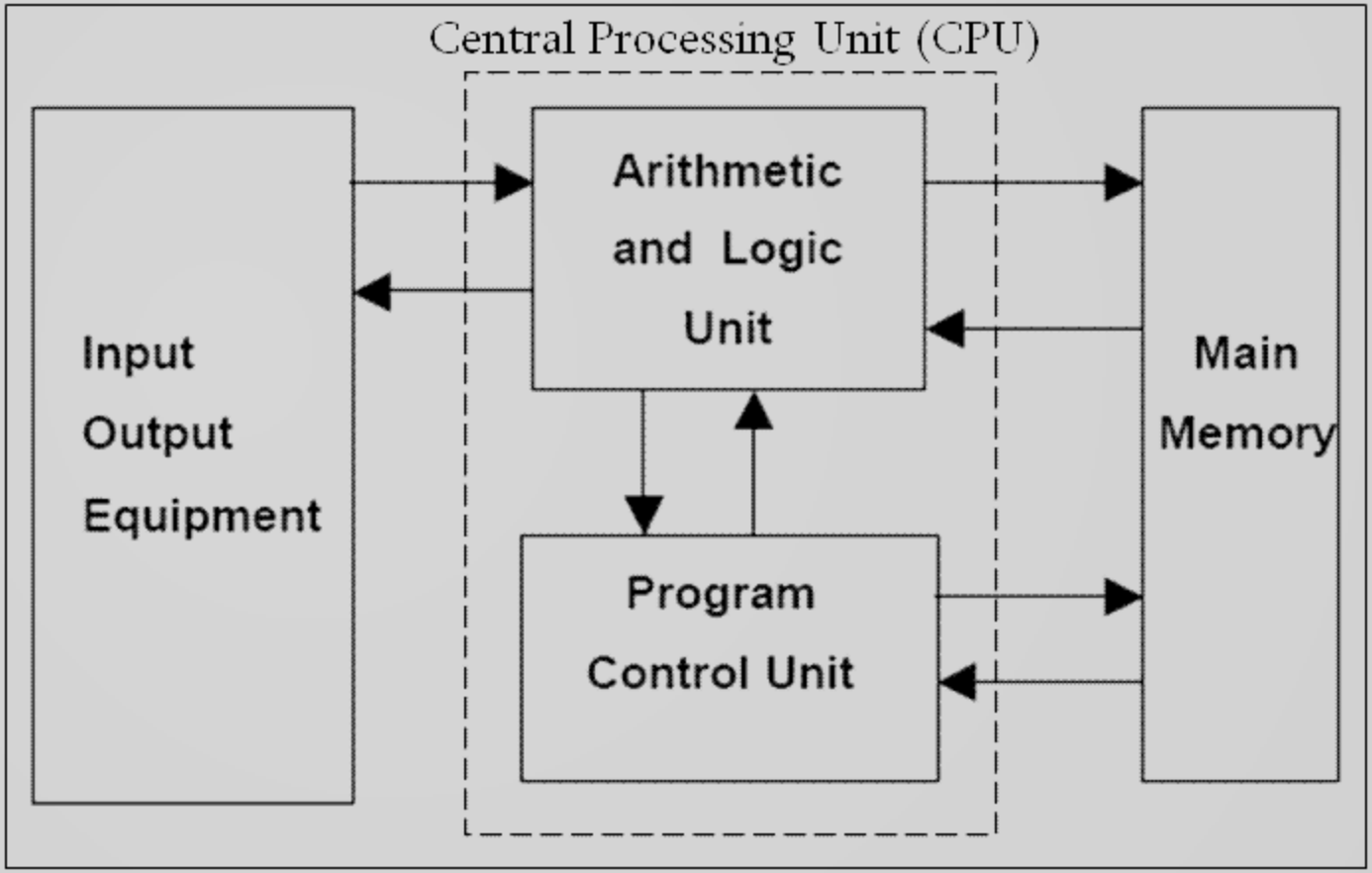
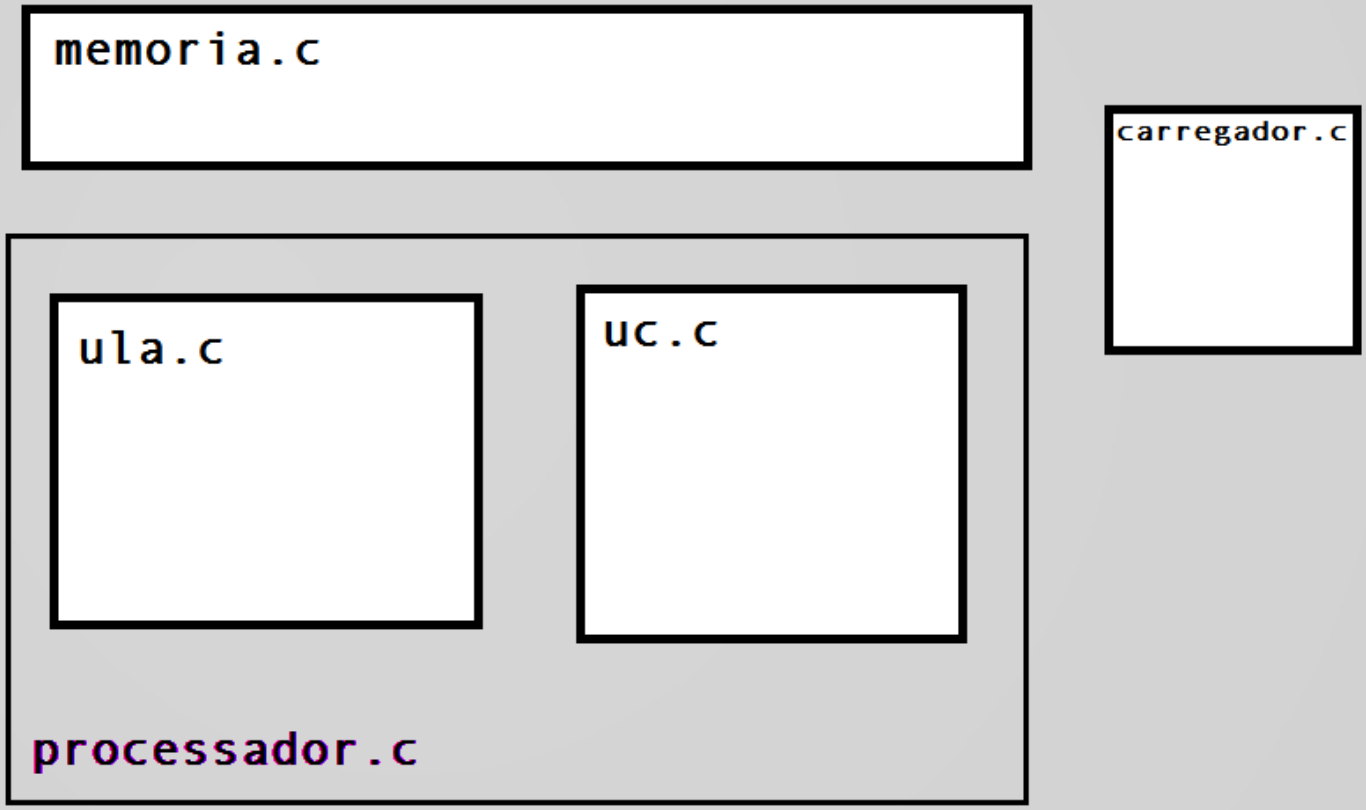


Figure : General structure of Von Neumann Architecture

Divisão da arquitetura IASMaS



Memoria.c

- Equivalente à “Memória” da arquitetura IAS original.
- Utilizada para guardar instruções e dados utilizados durante a execução.
- Possui as funções:
  - ZerarMemoria()
  - LerMemoria()
  - GravarMemoria()
  - GravarMem()

ZerarMemoria()

- Grava o valor zero em todos os 4096 endereços de memória.
- Utilizada apenas como um recurso de segurança.

LerMemoria()

- Efetua uma leitura na memória em um endereço informado pelo registrador MAR.
- Ao ser chamada, a função busca o dado no endereço informado e grava este valor em MBR.

GravarMemoria()

- Efetua uma gravação na memória em um endereço informado pelo registrador MAR.
- Ao ser chamada, a função copia o valor que está em MBR para o endereço informado.

GravarMem(x, valor)

- Recebe como entrada dois valores.
- O primeiro é um inteiro de 12 bits (sem sinal) correspondendo ao endereço desejado.
- O segundo é um inteiro de 40 bits (com sinal) correspondente ao valor que deve ser gravado.
- Esta função faz parte de uma versão antiga do sistema de gravação de dados, mas é utilizada durante o carregamento por ser mais eficiente.

ULA.c

- Equivalente à “ULA” da arquitetura IAS original.
- Responsável por operações aritméticas.
- Contém a definição dos registradores AC, MQ e MBR.
- Possui a função ULA().
- A função ULA possui código desenvolvido em assembly x86, sintaxe GAS.

ULA(x)

- Recebe como entrada um valor inteiro, que define a operação a ser realizada.
- Todos os operandos necessários devem estar nos registradores corretos no momento da chamada.
- As operações são identificadas a seguir.

- 0: Add m(x)
- 1: Add |m(x)|
- 2: Sub m(x)
- 3: Sub |m(x)|
- 4: Mul m(x)
- 5: Div m(x)
- 6: Lsh
- 7: Rsh
- 8: Load MQ m(x)
- 9: Load MQ
- 10: MBR -> AC
- 11: AC -> MBR

UC.c

- Equivalente à “UC” da arquitetura IAS original.
- Realiza o Ciclo de Instrução.
- Contém as definições dos registradores IR, IBR, MAR e PC.
- Possui as funções:
  - CicloBuscaEsquerda()
  - CicloBuscaDireita()
  - CicloExecucao()
  - Estatistica()

CicloBuscaEsquerda()

- Acessa um endereço na memória e copia todo o conteúdo para os registradores corretos.
- Prepara a parte esquerda deste endereço para a execução e guarda a direita em IBR.

CicloBuscaDireita()

- Busca a segunda metade do endereço de memória.
- Dividido em duas versões:
  - Se a instrução em IBR for a próxima
  - Se a instrução em IBR não for a próxima

CicloExecucao()

- Ocorre após cada ciclo de busca.
- Decodifica a instrução.
- Busca os operandos.
- Executa as operações necessárias, incluindo chamadas à ULA()

Estatistica()

- Exibe informações sobre as instruções executadas.
- É utilizada apenas ao final da execução.
- Não representa um conceito original do IAS.

Carregador.c

- De certo modo, pode ser considerado equivalente à “I / O” da arquitetura IAS original.
- Efetua o carregamento do programa IAS para a memória.
- Possui a função Carregador().

Carregador(caminho, nome)

- Recebe como entrada o caminho e o nome do arquivo.
- O caminho é relativo ao local do executável “iasmas”.
- O nome do arquivo não possui extensão.
- Estes dois dados são combinados para localizar e abrir o arquivo com extensão HEX, que foi gerado anteriormente por um Tradutor.
- O conteúdo do arquivo HEX é passado diretamente à memória, cada linha do arquivo corresponde a um endereço na memória do Iasmas.

Processador.c

- É onde ocorrem os processos relacionados ao fluxo de um programa IAS.
- Conceitualmente, inclui a UC e a ULA e por consequência os registradores destas unidades.
- No Iasmas, é responsável por ativar a função UC(), gerando um ciclo de execução.
- Possui as funções:
  - ZerarRegistradores()
  - Processador()

Explicação sobre o código que dá suporte à simulação

E OS MÓDULOS EXTRAS?

StdIAS.h

- O nome está relacionado à Standard IAS (Padrão IAS).
- Contém definições de novos tipos de dados:
  - ***int40_t***: Um inteiro com 1 bit que representa o sinal e 39 bits que representam o valor;
  - ***uint20_t***: Um inteiro com 20 bits que representam um valor sem sinal;
  - ***uint12_t***: Um inteiro de 12 bits sem sinal.
- Estes tipos foram utilizados para criar os registradores, e em momentos onde era necessário uma transferência de valores mais fiel ao IAS original.

Registradores.h

- Declara os registradores como variáveis externas.
- Deste modo, os registradores ficam disponíveis para todas as unidades de compilação que possuírem este header.
- É importante lembrar que, os registradores foram apenas declarados aqui, as definições estão em suas respectivas unidades (ULA ou UC).

Info.c

- Responsável pela exibição e/ou registro de informações durante a execução.
- Possui as funções:
  - PrintRegs() – Exibe alguns registradores na tela;
  - PrintRegsAll() – Exibe todos os registradores na tela;
  - LogRegs(arquivo) – Salva todos os registradores em texto;
  - PrintMem(faixa) – Exibe uma faixa de memória na tela;
  - LogRegs(arquivo, faixa) – Salva uma faixa de memória;

Tradutor.c

- Responsável pelo processo de tradução (ou conversão) das instruções IAS e valores em números hexadecimais correspondentes.
- Possui as funções:
  - RemoverEspacos()
  - ReconhecerTipo()
  - GerarNumero()
  - ObterInstrucaoPura()
  - ObterInstrucao()
  - Tradutor()

RemoverEspacos(linha)

- Recebe como entrada uma linha de texto.
- Procura e remove todos os espaços e quebras de linha (\n).
- Também converte as letras para sua versão minúscula.

ReconhecerTipo(linha)

- Recebe como entrada uma linha de texto.
- Verifica o conteúdo da linha e retorna uma classificação:
  - 0: A linha contém algum erro;
  - 1: A linha contém um valor numérico decimal;
  - 2: A linha contém apenas a expressão <***code***>
  - 3: A linha contém apenas a expressão <***data***>
  - 4: A linha contém uma instrução que não necessita de um endereço de memória (Instrução pura).
  - 5: A linha pode conter uma instrução ou um erro não reconhecido.

GerarNumero(linha)

- Retorna um número inteiro correspondente ao número que está na string linha.
- É importante que a linha contenha um número decimal válido, e nada mais.

ObterInstruçãoPura(linha)

- Verifica qual é a instrução contida na linha.
- Retorna um valor inteiro de 20 bits que corresponde aos 8 bits do opcode com os 12 bits do endereço (zeros).

ObterInstrução(linha)

- Verifica qual é a instrução contida na linha.
- Procura pelo endereço contido dentro da instrução.
- Retorna um valor inteiro de 20 bits que corresponde aos 8 bits do opcode com os 12 bits do endereço.
- A implementação é bastante complexa.

Tradutor(caminho, nome)

- Recebe o caminho e o nome do arquivo IAS.
- Abre este arquivo e cria um arquivo HEX, de mesmo nome e no mesmo local do arquivo IAS.
- Para cada linha do arquivo IAS:
  - Remove os espaços;
  - Verifica o tipo do conteúdo;
  - Obtém um número (40 bits) correspondente a um valor ou um par de instruções;
  - Grava este número no arquivo HEX e quebra a linha;
  - Se encontrar erros, interrompe a tradução.

InterfaceSimulador.c

- É por onde o simulador inicia e termina.
- Todas as interações com o usuário ocorrem por aqui.
- Responsável por coletar opções passadas pelo terminal, e informar erros não relacionados à execução.
- Possui as funções:
  - ExtrairCaminho() – Extrai apenas o caminho do arquivo;
  - ExtrairNome() – Extrai apenas o nome do arquivo;
  - InterpretaAlvo() – Verifica se o arquivo informado possui extensão .ias e caminho, retorna um código.

- É aqui que o Processador() é chamado, iniciando a execução.

Instruções sobre como utilizar o simulador

COMO FAZ?

Desenvolvendo seu código

- Utilize um editor de texto simples, pois alguns editores salvam a formatação no texto.
- Desenvolva normalmente utilizando as instruções do IAS no modo correto na qual são escritas.
- Escreva um número decimal, ou duas instruções por linha.
- Não se preocupe com espaços, eles serão removidos.
- Mantenha duas regiões distintas, as instruções e os dados deve estar sempre separados.
- Não há locais definidos para o tipo de informação, valores podem estar acima ou abaixo das instruções.

Desenvolvendo seu código

- Ao terminar, procure seus pares de instruções e separe-os.
- Deve haver apenas uma instrução por linha.

- Feito isto, adicione uma linha com a expressão **<code>** antes de sua primeira instrução.
- Adicione a instrução **exit** na linha abaixo de sua última instrução, ela marca o fim do programa.

- Novamente, não se preocupe com espaços ou linhas em branco.

Desenvolvendo seu código

- Agora encontre sua região de valores numéricos.
- Adicione uma linha com a expressão **<data>** antes do primeiro número.
- Salve o arquivo de texto com o nome que desejar e a extensão **.ias**
- Pronto, seu código está no formato correto para ser simulado.

Usando o simulador

- O simulador ainda está em fase de testes, e não possui instalador.
- Para utiliza-lo, é necessário:
  - Entrar na pasta do projeto “IAS Computer Simulator”;
  - Entrar na subpasta “bin”;
  - Entrar na subpasta “Debug”;
  - Abrir o terminal nesta pasta;
  - Executar “./iasmas -h”

Usando o simulador

- Ou pela linha de comando:
  - **cd ~/LOCAL_DO_DOWNLOAD**
  - **cd “IAS Computer Simulator”**
  - **cd bin**
  - **cd Debug**
  - **./iasmas -h**

Usando o simulador (-h)

- Mostra a ajuda de forma resumida.

Usando o simulador (-s)

- Não exibe informações durante a execução.
- Tem prioridade sobre as outras opções, com exceção de -l.

Usando o simulador (-r)

- Com esta opção ativa, todos os registradores serão mostrados a cada ciclo.
- Isto inclui os registradores MBR, AC, MQ, IR, IBR, MAR e PC.

- Se utilizado em conjunto com –s, esta opção é desativada.

Usando o simulador (-p)

- Indica uma partição (ou faixa) de memória a ser monitorada a cada ciclo.

- Devem ser definidos o primeiro e último valor a serem monitorados. Separados por uma "/".
- Ex: *./iasmas –p 10/21*

- Se utilizado em conjunto com -s, a faixa de memória só será mostrada ao final da execução.
- Se utilizado em conjunto com -l, a faixa de memória também será gravada no arquivo LOG.

Usando o simulador (-l)

- Cria um arquivo de texto com extensão LOG, onde serão armazenadas informações sobre a execução.

- Por padrão, serão salvos os registradores a cada ciclo.
- Se utilizado em conjunto com o -p, mostra também o conteúdo de uma faixa de memória a cada ciclo.

- Não é afetado por –s, -h, -r, -m.

Usando o simulador (-m)

- Limita o endereço máximo de memória que pode ser acessado.
- Esta função ainda não foi implementada.

Informações gerais sobre o projeto IASMaS

MAS QUE NOME É ESSE?

IAS Machine Simulator

- É um simulador em software do computador IAS, desenvolvido por John Von Neumann de 1945 a 1951.
- O ***Iasmas*** foi desenvolvido por Marlon Dias, entre Outubro e Novembro de 2013.
- Foi utilizada a linguagem C, com algumas partes em Assembly x86 (sintaxe GAS).

ENJOY YOUR SIMULATION