**Disciplina**: Performance em Sistemas Ciberfisicos

**Professor:** Guilherme Schnirmann

**Nome Estudante: Enzo Enrico Boteon Chiuratto**

**Atividade Prática / Relatório**

**Computador IAS**

# **Descrição da Atividade:**

O objetivo dessa atividade é entender como funciona o computador IAS. Esse computador é o primórdio dos computadores atuais, ou seja, é um computador Von Neumann, ainda que com estrutura arcaica é uma excelente ferramenta para entender os fundamentos e características do processador.

A memória do computador IAS é dividida em 4096 palavras (4k = 2¹²). Ou seja, temos uma memória com 12 bits de endereçamento. No nosso simulador o endereçamento está sendo feito em hexa, ou seja, 4 bits para cada dígito. Exemplo:

Posição 0 = 0000 0000 0000 = 000

Posição 10 = 0000 0000 1010 = 00A

Posição 1000 = 0011 1110 1011 = 3EB

Vamos utilizar um simulador desenvolvido na UNICAMP:

Estamos no nível mais baixo da arquitetura, ou seja, aqui as instruções são codificadas em linguagem de máquina. O formato da instrução da arquitetura do computador IAS (em hexadecimal):

**000** **01** **001** **05** **002**

**12 bits 20 bits 20 bits**

**Endereço instrução instrução**

**na memória**

Repare que temos os 3 primeiros dígitos representando o endereço em que as 2 próximas instruções serão armazenadas ao mapear em memória. **Cada dígito é um hexa e representa 4 bits.**

**000** **01** **001** **05** **002**

= **0000 0000 0000** **0000 0001** **0000 0000 0001** **0000 0101** **0000 0000 0010**

**0 0 0** **0 1** **0 0**  **1** **0 5** **0 0 2**

**Mapa memória opcode endereço opcode endereço**

**Atenção: os 3 primeiros dígitos não fazem parte da instrução! Lembre-se que a instrução tem 40 bits (começa no primeiro opcode).**

A seguir algumas instruções (opcodes) básicas:

**LOAD (01): carrega valor do endereço de memória no AC: AC <-M(X)**

**STOR (21); escreve valor do AC no endereço de memória M(X) <- AC**

**ADD (05); soma valor do endereço de memória no AC: AC <- AC +M(X)**

**SUB (06); subtrai valor do endereço de memória no AC: AC <- AC -M(X)**

**MUL (0B); multiplica valor do endereço de memória no MQ: <- MQ\*M(X)**

**LOAD MQ (mem.) (09); Carrega valor da memória para MQ: MQ <-M(X)**

**LOAD MQ AC (0A); Carrega valor de MQ para AC: AC<-MQ**

**DIV (0C). Divide valor de AC por valor de endereço da memória e resultado vai para MQ e resto para AC: MQ<-AC/M(X)**

**M(X) é o endereço que será o “parâmetro” na instrução do opcode.**

Para utilizar o simulador, deve-se atribuir na memória as instruções em hexadecimal. **Exemplo**:

**005 00 000 00 002** (valor atribuído em memória no endereço 005) **-DADO**

**006 3** (valor atribuído em memória no endereço 006) **-DADO**

**000 01 005 05 006;**

**- 000**: endereço de atribuição em memória (mapeamento)

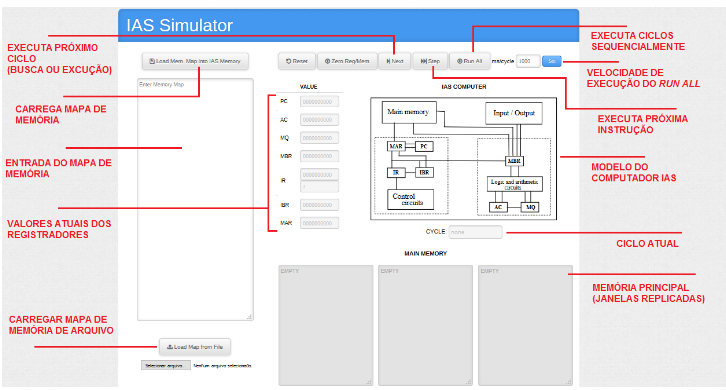
- **01**: instrução **LOAD** em hexadecimal;

- **005**: Endereço de memória de que vai ser feito o LOAD

-**05**: instrução ADD em hexadecimal

- **006**: endereço de memória de que vai ser feito o ADD

**Simulador:**



**Entrega:**

Esta atividade deverá ser entregue até o final da aula no Canvas.

O estudante deverá entregar um arquivo “.pdf” contendo as respostas da atividade proposta no roteiro.

**Roteiro da Atividade:**

1. Acesse o simulador IAS: <https://www.ic.unicamp.br/~edson/disciplinas/mc404/2017-2s/abef/IAS-sim/>
2. Considerando o exemplo inicial:

000 01 008 05 009

008 00 000 00 002

009 00 000 00 003

Copie o código em “Enter Memory Map” e clique em “load Mem. Map into IAS Memory”. **Sempre que fizer uma alteração no seu código, você vai precisar resetar os registradores e carregar novamente o mapa de memória.**

1. Execute passo-a-passo (clicando em **next**) e coloque um print de cada passo executado.
   1. Explique e registre o que cada registrador está recebendo/acessando em cada passo.

Diagrama

Descrição gerada automaticamente

Instruções carregadas na memória principal, program counter se inicia na linha 00

Interface gráfica do usuário

Descrição gerada automaticamente com confiança baixa

O valor no endereço de memória 008 (2 decimal) é carregado no acumulador, program counter se mantém na linha 0

Diagrama

Descrição gerada automaticamente

Program counter pula para o valor 001, com IR apontando para a próxima instrução (05 009)

Uma imagem contendo Tabela

Descrição gerada automaticamente

Instrução 05 009 é executada, adicionando (instrução 05) o valor do endereço de memória 009 (3 decimal) ao acumulador

Diagrama

Descrição gerada automaticamente com confiança baixa

Nenhuma instrução pendente para ser executada

* 1. Adicione uma linha no programa que armazene o resultado da soma no endereço 007. Apresente o print da memória.

**001 21 007 00 00** (Na linha 001, IR 21(stor m(x)) adiciona o valor do acumulador ao endereço 007)

Tabela

Descrição gerada automaticamente

1. Considere o seguinte exemplo:

000 01 005 0b 006

001 21 007 00 000

005 00 000 00 002

006 00 000 00 003

* 1. Antes de simular, traduza esse código de máquina, o que está sendo feito e qual o resultado esperado?

Na linha 000, o valor do endereço 005 (2 decimal) é adicionado ao acumulador. Multiplica o valor de MQ pelo valor do endereço 0006 (3 decimal).

Na linha seguinte, carrega o valor do acumulador ao endereço de memória 007, porém o acumulador é zerado após a operação 0b, assim atribuindo 0 ao valor 007

* 1. Execute passo-a-passo e explique o que está acontecendo. Você chegou no resultado esperado? Dica: Tanto a soma como a subtração são realizadas no registrador AC; Na divisão e multiplicação utilizamos o registrador MQ ( de formas diferentes )

Uma imagem contendo Diagrama

Descrição gerada automaticamente

O programa se inicia com o PC na linha 000 e o IR apontando para a primeira instrução (load)

Uma imagem contendo Diagrama

Descrição gerada automaticamente

A instrução 01 (load) é executada, trazendo o valor no endereço de memoria 005 (2 decimal) para o acumulador

Diagrama

Descrição gerada automaticamente

O program counter segue para a próxima instrução, com o IR apontando para ela

Uma imagem contendo Interface gráfica do usuário

Descrição gerada automaticamente

O valor do endereço 006 (3 decimal) é multiplicado pelo valor no registrador MQ, como o registrador está vazio, o valor retornado é 0 no endereço 007

Diagrama

Descrição gerada automaticamente

O valor do acumulador é escrito no endereço de memória 007

* 1. Agora execute o seguinte código – compare com o código inicial e explique.

000 09 005 0b 006

001 0a 000 21 007

005 00 000 00 002

006 00 000 00 003

O código agora armazena o valor da multiplicação no endereço 007, isso se dá a função 01 na linha 000 ter sido trocado pela função 09, onde ao invés de armazenarmos o valor no acumulador, o armazenamos diretamente no registrador MQ

1. Traduza em linguagem de máquina e execute no simulador (coloque os prints).

0: LOAD 500, ADD 501

1: STOR 500

500: 2

501: 5

000 01 1f4 05 1f5

001 21 1f4 00 000

1f4 00 000 00 002

1f5 00 000 00 005

Uma imagem contendo Gráfico de caixa estreita

Descrição gerada automaticamente

Iniciamos o programa com PC na linha 000

Interface gráfica do usuário

Descrição gerada automaticamente com confiança baixa

O valor do endereço de memoria 1f4 (500) é carregado para o acumulador

Uma imagem contendo Diagrama

Descrição gerada automaticamente

Program counter segue para a próxima instrução

Uma imagem contendo Diagrama

Descrição gerada automaticamente

O valor de 1f5 (501) é adicionado ao acumulador (2 + 5)

Diagrama

Descrição gerada automaticamente com confiança média

Program counter segue para a próxima instrução

Interface gráfica do usuário

Descrição gerada automaticamente com confiança média

O valor do acumulador é carregado ao endereço de memoria 1f4, subscrevendo o valor previamente armazenado

1. Divisão:
   1. Faça a divisão de 6 por 2. O resultado fica em qual registrador?

O resultado fica no registrador MQ, e o resto da divisão (nesse caso 0) se mantém no acumulador

* 1. Armazene o resultado anterior em algum endereço de memória. Que linha de instruções foi adicionada?

001 0A 000 21 00A

(Armazenar o valor do registrador MQ para AC e então o valor em AC para o endereço 00A)

* 1. Faça a divisão de 7 por 2. Como fica o resultado agora? Onde fica o resto?

O resultado em MQ fica 6, no acumulador é adicionado o resto 1.

* 1. Agora faça a conta (6/2) + 4 e armazene o resultado em algum endereço de memória. O que é preciso fazer para realizar a soma? (Dica: exercício 4) Execute passo-a-passo e coloque os prints explicando cada um.

00 01 0F1 0C 0F2

01 0A 000 05 0F3

0F1 00 000 00 006

0F2 00 000 00 002

0F3 00 000 00 004

Diagrama

Descrição gerada automaticamente

­­Programa se inicia com PC na linha 00

Interface gráfica do usuário

Descrição gerada automaticamente com confiança média

O valor de 0F1 é adicionado ao acumulador (AC = 6)

Diagrama

Descrição gerada automaticamente

Program counter avança

Interface gráfica do usuário

Descrição gerada automaticamente com confiança média

Realizamos a divisão do acumulador pelo valor em 0F2 (2 decimal), assim o acumulador é zerado por não termos nenhum resto e MQ se torna MQ = 3

Uma imagem contendo Diagrama

Descrição gerada automaticamente

Program counter avança

Diagrama

Descrição gerada automaticamente

O valor de MQ é carregado para o acumulador

Uma imagem contendo Diagrama

Descrição gerada automaticamente

PC avança

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

O valor do endereço 0F3 (4 decimal) é adicionado ao acumulador (previamente 3 decimal, agora, 7 decimal)

1. Conclusão: explique o que é cada um dos registradores e qual o fluxo de movimentação de dados e instruções no computador IAS. Existem diferença em operadores utilizados para soma/subtração e divisão/multiplicação? Se você já explicou em outros itens, replique aqui ou cite onde foi explicado).

O acumulador é o registrador principal para armazenar somas e subtrações, serve também como uma espécie de cache intermediário para escrever e ler valores não necessariamente endereçados na memória

O MQ serve como o Acumulador, porém usado principalmente em operações de multiplicação e soma, agindo também como um “cache intermediário” dentre operações, porém não sendo usado por funções (com exceção de funções que usam o valor de MQ para subscrever o acmulador)

O Registrador IR aponta para a próxima instrução a ser executada no começo de um ciclo de clock.

O registrador IBR armazena a próxima instrução e endereço de memória da instrução a ser executada no ciclo atual

O registrador MAR armazena qual endereço de memória será usado na operação a ser executada no ciclo atual

O PC (Program counter) mantém track de qual linha de instrução na memória está sendo executado atualmente.