**Disciplina**: Performance em Sistemas Ciberfisicos

**Professor:** Guilherme Schnirmann

**Nome Estudante: Icaro Lima Kuchanovicz**

**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)**

**JUMP M(X, INSTRUÇÃO ESQUERDA) – (0D) – O Program Counter salta para a instrução à esquerda da palavra na memória armazenada no endereço M(X).**

**JUMP M(X, INSTRUÇÃO DIREITA) – (0E) – O Program Counter salta para a instrução à direita da palavra na memória armazenada no endereço M(X).**

**JUMP+ M(X, INSTRUÇÃO ESQUERDA) – (0F) – Se AC>=0 então PC <-M(X). Salta para a instrução à esquerda da palavra de memória se o valor armazenado em AC for maior ou igual a zero.**

**JUMP+ M(X, INSTRUÇÃO DIREITA) – (10) – Se AC>=0 então PC <-M(X). Salta para a instrução à direita da palavra de memória se o valor armazenado em AC for maior ou igual a zero.**

**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)

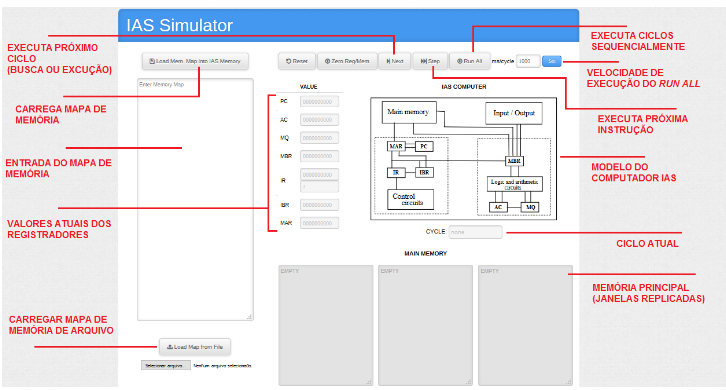
- **01**: instrução **LOAD** em hexadecimal;

- **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: Nessa atividade vamos conhecer as estruturas de salto dentro da memória principal. Fique atento aos comandos de JUMP adicionados na nossa lista no começo do arquivo:**

**JUMP M(X, INSTRUÇÃO ESQUERDA) – (0D) – O Program Counter salta para a instrução à esquerda da palavra na memória armazenada no endereço M(X).**

**JUMP M(X, INSTRUÇÃO DIREITA) – (0E) – O Program Counter salta para a instrução à direita da palavra na memória armazenada no endereço M(X).**

**JUMP+ M(X, INSTRUÇÃO ESQUERDA) – (0F) – Se AC>=0 então PC <-M(X). Salta para a instrução à esquerda da palavra de memória se o valor armazenado em AC for maior ou igual a zero.**

**JUMP+ M(X, INSTRUÇÃO DIREITA) – (10) – Se AC>=0 então PC <-M(X). Salta para a instrução à direita da palavra de memória se o valor armazenado em AC for maior ou igual a zero.**

Atente-se para copiar o código em “Enter Memory Map” e clicar 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.**

**Atenção: só coloque os prints que forem necessários para explicar o entendimento da lógica, se conseguiu explicar com suas palavras o que está sendo feito, um print com resultado de funcionamento está suficiente.**

1. Acesse o simulador IAS: <https://www.ic.unicamp.br/~edson/disciplinas/mc404/2017-2s/abef/IAS-sim/>
2. Implemente o exercício da aula passada utilizando funções. Explique o fluxo de alterações que no PC (Program counter). Coloque os prints dos resultados finais e jumps.

#Código

000 0D 2F1 21 010

001 0D 3F1 21 011

002 0D 1F1 21 012

#Função 1 - Soma

1F1 01 010 05 011

1F2 0E 002 00 000

#Função 2 - Divisão

2F1 01 00A 0C 00B

2F2 0A 000 0E 000

#Função 3 - Multiplicação

3F1 09 00C 0B 00D

3f2 0A 001 0E 001

#Variáveis

00A 00 000 00 010

00B 00 000 00 004

00C 00 000 00 014

00D 00 000 00 005

Interface gráfica do usuário

Descrição gerada automaticamente

def soma(a,b):  #(0A1)

    return a+b

def div(c,d):   #(0B1)

    return c/d

def mul(c,d):   #(0C1)

    return c\*d

a = 16 #(00A)

b = 4 #(00B)

c = 20 #(00C)

d = 5 #(00D)

x = div(a,b)

y = mul(c,d)

z = soma(x,y)

1. Faça o mesmo para:

#Código

000 0D 0F1 21 011 #x

001 0D 1F1 21 012 #y

002 0D 2F1 21 013 #z

003 0D 3F1 21 014 #r

#Função 1 - Multi

0F1 09 00A 0B 00B

0F2 0A 000 0E 000

#Função 2 - Div

1F1 01 00C 0C 00A

1F2 0A 000 0E 001

#Função 3 - Soma

2F1 01 011 05 012

2F2 0E 002 00 000

#Função 4 - Sub

3F1 01 013 06 011

3F2 0E 003 00 000

#Variáveis

00A 00 000 00 005

00B 00 000 00 003

00C 00 000 00 014

00D 00 000 00 004

Interface gráfica do usuário

Descrição gerada automaticamente com confiança média

def soma(a,b):  #(0A1)

    return a+b

def div(c,d):   #(0B1)

    return c/d

def mul(c,d):   #(0C1)

    return c\*d

def sub(c,d):   #(0D1)

    return c-d

a = 5

b = 3

c = 20

d = 4

x = mul(a,b)

y = div(c,a)

z = soma(x,y)

r = sub(z,x)

1. Implemente um contador de 10 passos no computador IAS. Tire um print do seu código e de uma posição em memória com o valor final do contador. Ou seja, ao final, você precisará ter em uma posição de memória o valor 10 (caso tenha contado de 1 em 1).

Consegui fazer de 2 jeitos

1)

#Código

000 01 00A 05 00B

001 21 00A 01 00C

002 06 00A 0F 000

003 21 00D 00 000

#Variáveis

00A 00 000 00 000 #soma

00B 00 000 00 001 #n

00C 00 000 00 00A #fim (i)

2)

#Código

000 01 00A 05 00D

001 21 00A 01 00C

002 05 00D 21 00C

003 01 00B 06 00C

004 0F 000 00 000

#Variáveis

00A 00 000 00 000 #soma

00B 00 000 00 00A #n

00C 00 000 00 001 #i

00D 00 000 00 001 #aux

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

1. Implemente o fatorial de um número no computador IAS.

#Código

000 09 00A 0B 00B

001 0A 000 21 00A

002 01 00C 05 00D

003 21 00C 01 00B

004 06 00C 0F 000

#Variáveis

00A 00 000 00 001 #fat

00B 00 000 00 005 #n

00C 00 000 00 001 #i

00D 00 000 00 001 #soma

Interface gráfica do usuário, Aplicativo, Tabela

Descrição gerada automaticamente

1. Implemente no computador IAS o seguinte somatório:

Interface gráfica do usuário, Aplicativo, Tabela

Descrição gerada automaticamente#for(i = 1; i <= 8; i++)

# 2 \* i + 11

###codigo inteiro

#soma

000 09 00D 0B 00B

001 0A 000 05 00E

002 05 00A 21 00A

#conferindo

003 01 00B 05 00F

004 21 00B 01 00C

005 06 00B 0F 000

#salvando

006 21 010 00 000

#variáveis

00A 00 000 00 000 #soma

00B 00 000 00 001 #i

00C 00 000 00 008 #n

00D 00 000 00 002 #aux1

00E 00 000 00 00B #aux2

00F 00 000 00 001 #aux3

1. Implemente no computador IAS uma verificação se um número é par ou ímpar e realize as operações apresentadas no código. Apresente o seu código no computador IAS, bem como, as variáveis na memória com seus valores finais. Faça o teste para um número par (ex: a = 10) e um número ímpar (ex: a = 11).

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente#código

000 01 00A 0C 00B

001 06 00C 0F 010

#if

002 09 00A 0B 00B

003 0A 000 21 00D

#else

010 01 00A 05 00B

011 21 00D 00 000

#variáveis

00A 00 000 00 00A

00B 00 000 00 002

00C 00 000 00 001

00D 00 000 00 000

a = 10

b = 2

c = 1

d = 0

if (a % 2 == 0):

    d = a\*b

else

    d = a+b

1. Faça um programa que que execute 10 passos. Deve-se verificar se o índice atual é par ou ímpar. Caso seja par some 5 em uma variável, caso seja ímpar some 3 na mesma variável. Utilize a estrutura de funções para verificar se é par ou ímpar e também para somar e subtrair.

#Código

000 01 00B 0C 010

001 06 00D 0F 0AA

002 01 00A 05 00F

003 21 00A 01 00B

004 05 00D 21 00B

005 01 00C 06 00B

006 0F 000 00 000

0AA 01 00A 05 00E

0AB 21 00A 10 003

#Variáveis

00A 00 000 00 000 #oi

00B 00 000 00 001 #i

00C 00 000 00 00A #n

00D 00 000 00 001 #aux1

00E 00 000 00 005 #par

00F 00 000 00 003 #impar

010 00 000 00 002 #div

Interface gráfica do usuário

Descrição gerada automaticamente