Pontifícia Universidade Católica do Paraná
Escola Politécnica
Bacharelado em Engenharia de Software

Disciplina: Performance em Sistemas Ciberfisicos
Professor: Guilherme Schnirmann

Nome Estudante: Ary Felipe Farah e Silva

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 = 2^{12}). 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):

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)

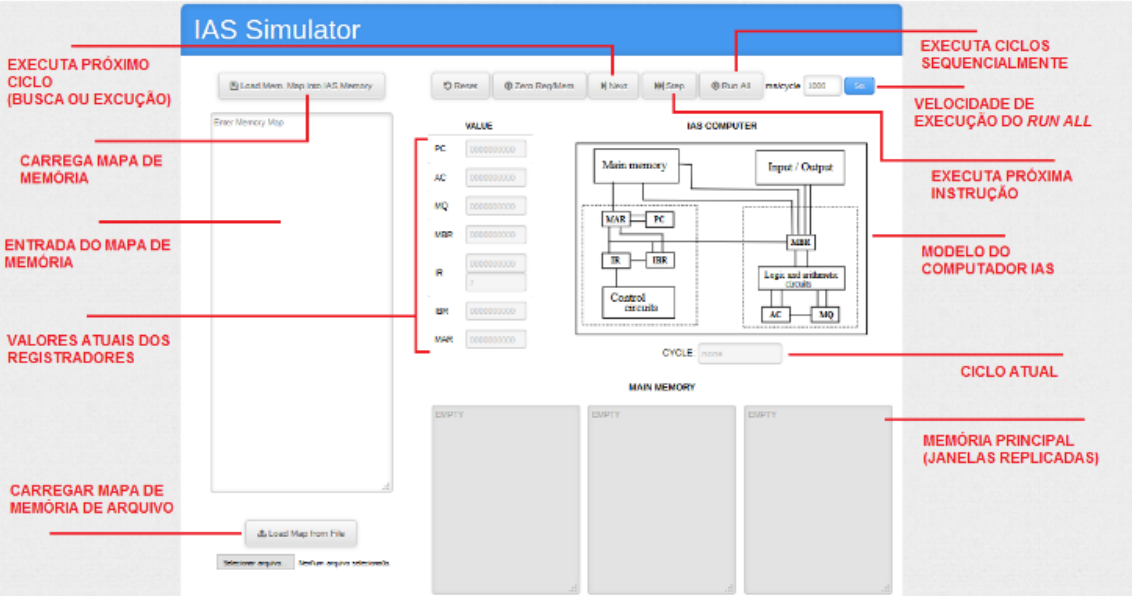
- 01: instrução LOAD em hexadecimal;

- 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. Revisando a prática anterior: Implemente no computador IAS (você escolhe as posições de memória de dados que irá utilizar).

```
a = 16
b = 4
c = 20
d = 5

x = a/b + c/d
```

CÓDIGO FEITO

000 01 00A 0C 00B

001 0A 000 21 0AA

002 01 00C 0C 00D

003 0A 000 21 0BB

004 01 0AA 05 0BB

005 21 00F 00 000

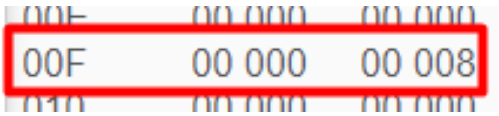
00A 00 000 00 010

00B 00 000 00 004

00C 00 000 00 014

00D 00 000 00 005

RESULTADO:



Foi necessário converter os valores para Hexadecimal, depois só definir os endereços de memória para cada valor e programar com eles

3. Implemente no computador IAS:

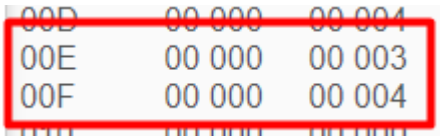
```
a = 5
b = 3
c = 20
d = 4

x = (a*b + c/a)/d
```

CÓDIGO FEITO

000 09 00A 0B 00B
001 0A 000 21 0AA
002 01 00C 0C 00A
003 0A 000 21 0BB
004 01 0AA 05 0BB
005 21 0CC 0C 00D
006 21 00E 0A 000
007 21 00F 00 000

00A 00 000 00 005
00B 00 000 00 003
00C 00 000 00 014
00D 00 000 00 004

RESULTADO:



00F = resultado da divisão
00E = resto da divisão

Foi necessário converter os valores para Hexadecimal, depois só definir os endereços de memória para cada valor e programar com eles

4. Considerando o exemplo:

000 01 10B 05 10C ***//000 LOAD M(10B) ; ADD M(10C)***

001 21 10B 0D 000 ***//001 STOR M(10B) ; JUMP M(000, ESQ.)***

10B 00 000 00 002 ***//DADO 002 CARREGADO EM MEMÓRIA NA POSIÇÃO 10B***

10C 00 000 00 001 ***//DADO 001 CARREGADO EM MEMÓRIA NA POSIÇÃO 10C***

Execute passo-a-passo (clicando em **next**) e explique com suas palavras o que está sendo executado e o fluxo das informações no computador IAS.

10B 00 000 00 002
10C 00 000 00 001

DADOS CARREGADOS NO ENDEREÇO DE MEMÓRIA (2 -> 10B e 1 -> 10C)

```
000 01 10B 05 10C
001 21 10B 0D 000
```

COMANDOS A SEREM EXECUTADOS:

000: 01 – Carrega o valor da posição 10B no registrador AC.

05 – Adiciona ao valor de AC o valor da memória 10C.

001: 21 – Salva o valor de AC em uma memória (10B)

0D – JUMP – Pula para alguma linha (000) e faz o IR ler o comando da esquerda (01...)

Assim, esse comando cria uma espécie de contador que inicia em 2 e, cada vez que se dá uma volta, soma-se 1 ao número da memória 10B.

5. Considere o seguinte exemplo:

   000 01 11D 06 11E
   001 21 11D 0F 000
   11D 00 000 00 005
   11E 00 000 00 001

   a. Indique em cada linha o que significa as informações em memória. É instrução? É dado? O que significa cada grupo de dígitos em cada linha?

      000 01 11D 06 11E – Instrução - Mapa Memória – OPCODE – Endereço Mem
      001 21 11D 0F 000 – Instrução - Mapa Memória – OPCODE – Endereço Mem

      11D 00 000 00 005 – Dado – Valor atribuído - Endereço Mem
      11E 00 000 00 001 – Dado - Valor atribuído - Endereço Mem

   b. Antes de simular, traduza esse código de máquina, o que está sendo feito e qual o resultado esperado? Explique com suas palavras o que é executado nesse programa

      Primeira linha: carregar valor presente no endereço 11D no registrador AC e subtrair dele o valor presente em 11E.
      Segunda linha: salvar o valor obtido da subtração no endereço 11D, depois pula e executa novamente o comando da primeira linha, enquanto o valor presente em AC ainda for maior ou igual que 0.

      Basicamente, esse programa subtrai 11D(x) - 11E(y) até que a o valor armazenado no AC seja menor que 0.

c. Como ficaria esse código em uma linguagem de alto nível? (qualquer linguagem).

```python
# PYTHON
a = 5
b = 1
while a >= 0:
    print(a, '-', b, '=', end=' ')
    a -= b
    print(a)
print('FIM DO PROGRAMA')
```

```
5 - 1 = 4
4 - 1 = 3
3 - 1 = 2
2 - 1 = 1
1 - 1 = 0
0 - 1 = -1
FIM DO PROGRAMA
```

6. Considere o seguinte exemplo. Os endereços 00A, 00B e 00C são nossas variáveis X, Y e Z, respectivamente.

```
000 01 00A 06 00B
001 0F 003 01 00B
002 21 00C 0D 004
003 01 00A 21 00C

00A 00 000 00 002 // X = 2
00B 00 000 00 003 // Y = 3
00C 00 000 00 000 // Z = 0
```

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

Primeira linha: X – Y

Segunda linha: Pularia para linha 003 caso o valor em AC fosse positivo, mas como não é, o programa continua rodando no próximo código: carrega o valor de Y no AC.

Terceira linha: Salva o valor de Y em Z e pularia para a linha 004, mas como ela não existe, termina o programa.

X – Y – resultado negativo, não deixa o comando 0F funcionar

Y = Z – carrega o valor de Y no AC e o salva no endereço de Z.

b. Explique com suas palavras o que é executado nesse programa
Se X – Y < 0, então Z = Y
Se X – Y >= 0, então Z = X

c. O que acontece se mudarmos os valores para X = 5 e Y = 3?

Primeira linha: X – Y

Segunda linha: Pula para linha 003 (quarta linha), é.

Terceira linha: Salva o valor de Y em Z e pularia para a linha 004, mas como ela não existe, termina o programa.

X – Y – resultado positivo, o comando 0F funciona

X = Z – carrega o valor de X no AC e o salva no endereço de Z.

d. Como ficaria esse código em uma linguagem de alto nível? (qualquer linguagem).

```python
# PYTHON
x = int(input('Digite o valor de 00A: '))
y = int(input('Digite o valor de 00B: '))
z = 0

if x - y < 0:
    z = y
else:
    z = x

print(f'00A: {x} \n'
      f'00B: {y} \n'
      f'00C: {z}')
```

```
Digite o valor de 00A: 2
Digite o valor de 00B: 3
00A: 2
00B: 3
00C: 3
```

```
Digite o valor de 00A: 5
Digite o valor de 00B: 3
00A: 5
00B: 3
00C: 5
```