Faculdade de Computação

Arquitetura e Organização de Computadores 1

Laboratório de Programação Assembly 1

Prof. Cláudio C. Rodrigues

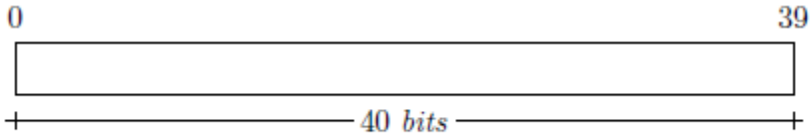
Programando a Arquitetura IAS Machine

O **IAS Machine** foi o primeiro computador eletrônico construído no Instituto de Estudos Avançados (daí o nome) em Princeton. O líder do projeto foi John Von Neumann, que também foi consultor do projeto ENIAC (o primeiro computador eletrônico de propósito geral). O projeto IAS foi muito importante porque foi um dos primeiros computadores a implementar o conceito de "programa armazenado", onde as instruções dos programas seriam armazenadas na memória, juntamente com os dados. Esse modelo facilitou muito o armazenamento e a edição de programas, e também possibilitou que o próprio programa fosse alterado em tempo de execução, facilitando o trabalho com desvios (*branches*) e arrays de dados.

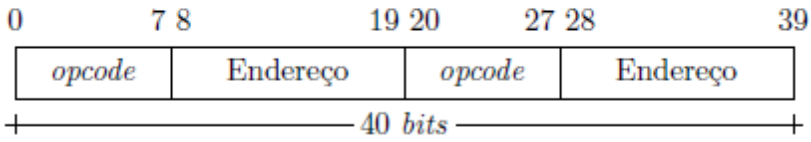
A arquitetura do IAS era muito simples, mas muito eficiente. Tanto que ficou conhecido como a arquitetura "Von Neumann" e é a base para praticamente todos os computadores modernos, incluindo o que você está usando no momento. A máquina IAS tinha 1024 endereços de memória, cada um com 40 bits de comprimento. Cada instrução ocupava 20 bits, onde os primeiros 8 bits eram o *opcode* e os 12 bits restantes eram o parâmetro de endereço.

A Arquitetura do Conjunto de Instruções contemplava apenas 20 instruções. O conjunto de instruções pode ser encontrado em arquivo anexo a esse documento.

A memória principal do Simulador do IAS possui 1024 palavras de 40 bits. Cada palavra está associada a um endereço distinto, um número que varia de 0 até 1023. Cada palavra da memória principal do Simulador do IAS pode armazenar um número de 40 bits ou duas instruções de 20 bits. Os números são representados em ***complemento de dois***. As instruções possuem dois campos, o "código da operação" de 8 bits, e o "endereço", de 12 bits. A Figura abaixo ilustra a representação de números e instruções em uma palavra de memória.



(a) Um número de 40 *bits* em uma palavra da memória



(b) Duas instruções de 20 *bits* em uma palavra da memória

Tarefa: Escreva sequências de códigos da Arquitetura IAS para resolver a lista de problemas a seguir. Construa o algoritmo em linguagem de montagem do IAS, traduza o código para a representação hexadecimal correspondente e simule a execução no aplicativo de simulação disponível no canal Softwares na equipe AOC1 BCC.

Instruções:

I. Apresentar as soluções usando a linguagem de montagem do IAS e codificação em hexadecimal.
II. O trabalho deve ser desenvolvido em grupo composto de 1 até 4 (um até quatro) estudantes e qualquer identificação de plágio sofrerá penalização;
III. Entrega dos resultados deverá ser feita por envio de arquivo resposta na plataforma MS Teams, impreterivelmente, no dia ***08/11/2020***.

Problemas:

P1. Considere o código de máquina abaixo, codificado para o computador IAS e armazenado em memória. Traduza esse código de máquina para a representação em linguagem de montagem *(**IAS assembly**}*, considere que o código começa no endereço 08A da memória. Descreva de forma objetiva o que este programa faz.

endereço	conteúdo
08A	010FA210FB
08B	010FA0F08D
08C	020FA210FB
08D	

P2. A potenciação ou exponenciação (**x^n**) é uma das operações básicas no universo dos números naturais onde um dado número **x** é multiplicado por ele mesmo, uma quantidade **n** de vezes. O fragmento 1 apresenta um algoritmo simples para calcular **x^n** (**x** elevado a **n**).

Fragmento 1

```
int expo1(int x, int n){
    int result = 1;
    while (n>0){
        result *= x;
        n--;
    }
    return result;
}
```

Embora esse algoritmo seja relativamente eficiente, com desempenho em tempo linear ou *O(n)*, ele pode ser aprimorado. Poderíamos fazer a mesma tarefa em *O(log (n) + log (n))*. De que maneira? Usando um método chamado ***exponenciação quadrática***.

Ideia básica: para qualquer **x^n**, se a **n** for par, poderíamos escrevê-lo como **(x ^ n / 2) ^ 2**. Se **n** for ímpar, por outro lado, poderíamos escrevê-lo como **x * (x ^ (n-1/2)) ^ 2**. Veja a figura abaixo:

$$x^n = \begin{cases} 1, & \text{if } n = 0 \\ \frac{1}{x}^{-n}, & \text{if } n < 0 \\ x \cdot \left(x^{\frac{n-1}{2}}\right)^2, & \text{if } n \text{ is odd} \\ \left(x^{\frac{n}{2}}\right)^2, & \text{if } n \text{ is even} \end{cases}$$

Uma versão iterativa do ***algoritmo de exponenciação quadrática*** é mostrado no Fragmento 2 onde, em cada etapa, divide-se o expoente por dois e eleva ao quadrado a base e, nas iterações em que o expoente é ímpar, você multiplica o resultado pela base.

Fragmento 2

```
int expo2(int x, int n){
  int result = 1;
  while (n){
    if (n%2==1){
      result *= x;
    }
    n /= 2;
    x *= x;
  }
  return result;
}
```

Tarefa: Escreva em linguagem de máquina do IAS os dois algoritmos (Fragmento 1 e Fragmento 2). Faça uma análise de desempenho dos dois algoritmos e descreva os resultados obtidos.

P3. Escreva em linguagem de máquina do IAS um programa que faça a classificação em pares ou ímpares de valores armazenados em um vetor **V** localizados em memória a partir da posição 100. O programa deve armazenar os valores classificados como pares no vetor "pares" localizado na posição de memória 110 e os valores classificados como ímpares no vetor "impares" localizado na posição de memória 120.

Fragmento 3

```
int main( ){
     int V[10], pares[10], impares[10];
     int i, j=0, k=0;

     for(i=0;i<10;++i) {
          if (V[i]%2==0) {
               pares[j] = V[i];
               j = j + 1;
          }else {
               impares[k] = V[i];
               k = k + 1;
          }
     }
   return 0;
}
```

Tarefa: Escreva sequências de códigos da Arquitetura IAS para resolver o problema de classificação (Fragmento 3). Construa o algoritmo em linguagem de montagem do IAS, traduza o código para a representação hexadecimal correspondente e simule a execução no aplicativo de simulação

==========

Exemplo de código gerador dos 10 primeiros valores da sequência de Fibonacci

000	JUMP M(007,0:19)	NOP	000 0D 00 70 00 00
001	LOAD M(097)	ADD M(098)	001 01 09 70 50 98
002	STOR M(099)	STOR M(100)	002 21 09 92 11 00
003	LOAD M(094)	ADD M(095)	003 01 09 40 50 95
004	STOR M(094)	STOR M(002,28:39)	004 21 09 41 30 02
005	LOAD M(098)	STOR M(097)	005 01 09 82 10 97
006	LOAD M(099)	STOR M(098)	006 01 09 92 10 98
007	LOAD M(096)	SUB M(095)	007 01 09 60 60 95
008	STOR M(096)	JUMP+ M(001,0:19)	008 21 09 60 F0 01
009	JUMP M(009,0:19)		009 0D 00 90 00 00
int main() {			094 00 00 00 01 00
int a = 0, b = 1, c, n = 10, i = 0x100, Valor_1 = 1;			095 00 00 00 00 01
while(i<n){			096 00 00 00 00 0A
c = a + b;			097 00 00 00 00 00
a = b;			098 00 00 00 00 01
b = c;			099 00 00 00 00 00
fibo[i] = c;			100 00 00 00 00 00
i = i + 1;			
} }			