Faculdade de Computação

Arquitetura e Organização de Computadores 1

Laboratório de Programação Assembly 1

Prof. Cláudio C. Rodrigues

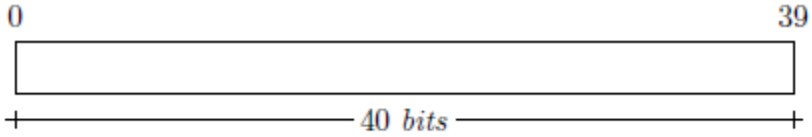
Programando a Arquitetura IAS Machine

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 possibilitou que o próprio programa fosse alterado em tempo de execução, facilitando o trabalho com desvios (*branches*) e arrays de dados.

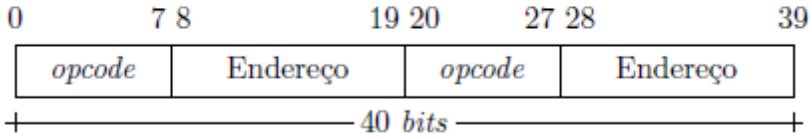
A arquitetura do IAS era muito simples, mas muito eficiente. Tanto que ficou conhecido como a arquitetura "Von Neumann" e é a base para praticamente todos os computadores modernos, incluindo o que você está usando no momento. A máquina IAS tinha 1024 endereços de memória, cada um com 40 bits de comprimento. 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 utilizam o formato de 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 da equipe AOC1 na plataforma MS Teams.

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 5 (um até cinco) discentes e qualquer identificação de plágio sofrerá penalização;
III. Entrega dos resultados deverá ser feita por envio de arquivo zipado com os seguintes artefatos de software: Memorial descritivo das soluções em pdf; arquivo em txt com os códigos que solucionam os problemas propostos em assembly e codificado em hexadecimal.
IV. Submeter os documentos na plataforma MS Teams, impreterivelmente, no dia ***28/08/2024***.

Problemas:

P1. *Optmization*: Um grupo de alunos deseja saber a somatória de idades de seus colegas, exceto a sua idade. Para solucionar o problema propuseram criar um vetor coleção de idades, onde cada elemento desse vetor armazena a idade de um determinado aluno. Um segundo vetor armazenaria o somatório das idades de seus colegas, exceto a sua idade. Escreva em linguagem de montagem (assembly) da *arquitetura IAS Machine* um programa que leia da memória um número inteiro positivo **n**, referentes ao tamanho da coleção de idades. O algoritmo deve acessar o vetor coleção de idades e armazenar num vetor resultante o somatório das idades de seus pares (colegas). Assim, os elementos do novo vetor serão atualizados da seguinte forma: o valor do elemento da posição de índice i será o somatório de todos os outros elementos, exceto o elemento da posição i.

Exemplo:

idades[] = {10, 4, 1, 6, 2}
somas[] = {13,19,22,17,21}

Tarefa: ***Escreva duas sequências de códigos*** da Arquitetura IAS para resolver o problema descrito, escreva um algoritmo com uma lógica de programação ruim (péssimo desempenho) e, outro algoritmo com uma lógica otimizada. Para avaliar o desempenho utilize a métrica da quantidade de instruções executadas.

Para simplificar e uniformizar a codificação, considere que as variáveis escalares necessárias para a solução do problema estarão armazenadas a partir do endereço de memória $\boldsymbol{020_{16}}$ e os vetores ***idades*** e ***somas*** nos endereços $\boldsymbol{030_{16}}$ e $\boldsymbol{050_{16}}$, respectivamente. 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 IAS Machine.

P2. *Program Challenge*: A ***conjectura de Collatz*** é uma conjectura matemática que recebeu este nome em referência ao matemático alemão Lothar Collatz, que foi o primeiro a propô-la, em 1937. Além desse nome, este problema também é conhecido como ***Problema 3x + 1***.

Esta conjectura se aplica a qualquer número natural, determinando o seguinte: se um número for par, dividi-lo por **2**, e se for ímpar, multiplicá-lo por **3** e adicionar **1** (**3x+1**). Repetir esse processo até que o número se torne 1. Desta forma, por exemplo, se a sequência iniciar com o número 5, teremos: 5; 16; 8; 4; 2; 1. A conjectura apresenta uma regra dizendo que, qualquer número natural, quando aplicado a esta regra, eventualmente sempre chegará a 4, que se converte em 2 e termina em 1.

Construa um programa em linguagem de montagem (assembly) da arquitetura *IAS Machine* que leia da memória um número natural **n** e escreva em endereços consecutivos da memória do IAS a sequência de Collatz.

Considere a seguinte operação em um número arbitrário **n** (inteiro positivo):

- Se o número **n** for par, divida-o por **2**;
- Se o número **n** for ímpar, multiplique-o por **3** e some **1 (3x+1).**

Exemplo:
n ? 6
sequência: 6 3 10 5 16 8 4 2 1

Tarefa: Para simplificar e uniformizar a codificação, considere que as variáveis escalares necessárias para a solução do problema estarão armazenadas a partir do endereço de memória $\boldsymbol{020_{16}}$ e os valores da ***sequência*** a partir do endereço $\boldsymbol{030_{16}}$. Escreva sequências de códigos da Arquitetura IAS para resolver o problema descrito. 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 IAS Machine.

P3. *Optmization*: 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 montagem do IAS os dois algoritmos (Fragmento 1 e Fragmento 2). Faça uma análise de desempenho dos dois algoritmos e descreva os resultados obtidos. Para facilitar a análise, contabilize o número de instruções executadas para um valor de n grande.

P4. *Program Challenge*: 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.

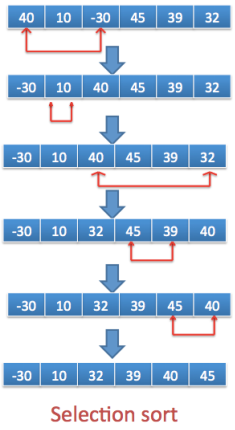
Fragmento 3	

`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 IAS Machine.

P5. *Sorting*: Escreva em linguagem de montagem do IAS um programa que realize a ordenação de elementos de um vetor **V** de inteiros, em ordem crescente. A ordenação deve ser feita no próprio vetor **V**, sem utilizar um vetor auxiliar. Considere que os elementos do vetor **V** estão armazenados a partir da posição 100_{16} da memória.

```
int main(){
 int n = 10, i, j, aux;
 int v[10] = {3,7,5,8,0,1,6,9,4,2};

 for(i=0; i<n-1; i++){
     for(j=i+1; j<n; j++){
         if(v[j]<v[i]) {
             aux = v[i];
             v[i] = v[j];
             v[j] = aux;
         }
     }
 }
}
```



Selection sort

P6. Escreva em linguagem de montagem do IAS um programa que calcule o valor do polinômio $\mathbf{p(x)=a_0+a_1x+...+a_nx^n}$ em **k** pontos distintos (valores de **x**). O programa receberá os valores de **n** (ordem do polinômio), dos coeficientes reais do polinômio ($\mathbf{a_0, a_1, ..., a_n}$), a quantidade de pontos **k** para o cálculo de **p(x)** e os **k** pontos de teste ($\mathbf{x_1, x_2, ..., x_k}$). Dica: um polinômio de ordem **n** pode ser representado pelos coeficientes guardados em um vetor de tamanho **n+1** elementos.

```
// p(x)= 88-27x- 7x^2 - 11x^3 + 2x^4
int main() {
  int i, j, k = 8, n = 4;
  int coef[5]={88,-27,-7,-11,2};
  int x[8] = {0,1,2,3,4,5,6,7};
  int Px[8]={0};
  for(i=1;i<=k;i++){
       for(j=0;j<=n;j++) Px[i] += coef[j]*pow(x[i],j);
  }
}
```