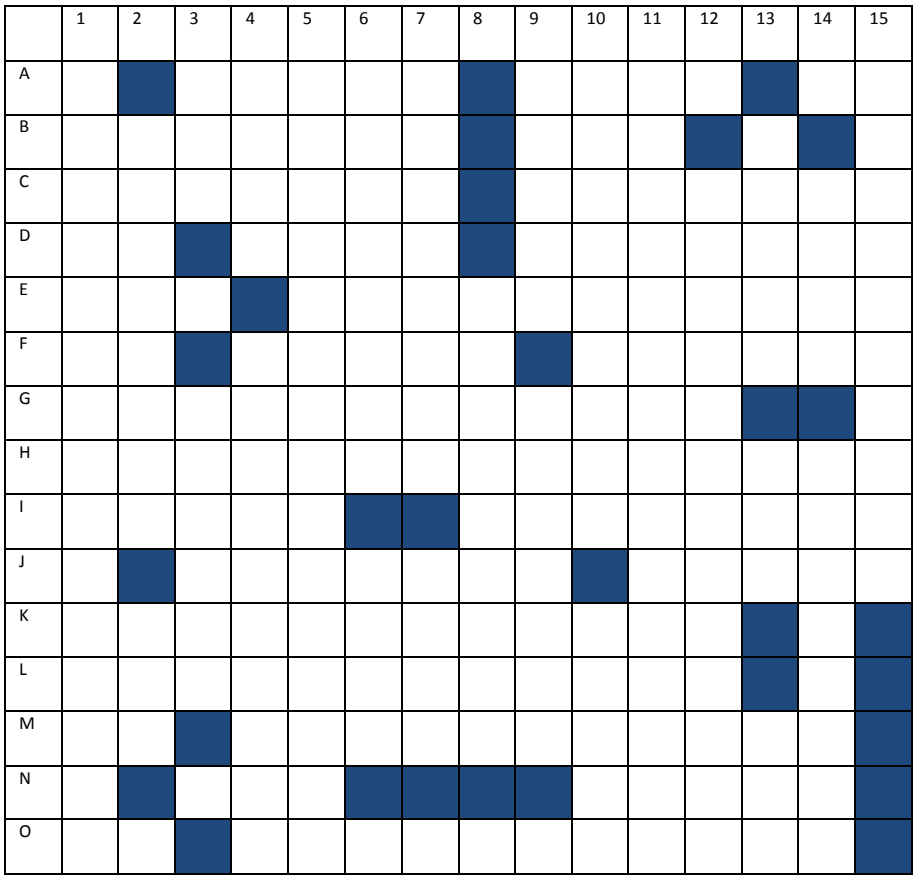
Instituto de Computação
Departamento de Ciência da Computação
Programação Estruturada
Professor: Anselmo Montenegro
26 de setembro de 2016

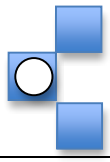
Primeira Lista

1) Implemente em C o jogo Pedra-Papel-Tesoura. Faça o usuário jogar contra o computador que deve selecionar o objeto de forma aleatória. A cada jogada a interface como o usuário pergunta se ele quer continuar o terminar.

2) A Batalha Naval é um jogo de tabuleiro muito popular, jogado usando lápis e papel. Ele consiste basicamente em destruir os navios e aeronaves do jogador adversário antes que o adversário destrua as suas. O ambiente do jogo é descrito por um tabuleiro (um reticulado) de **n** x **m** células, onde são posicionadas as partes das figuras correspondentes as armas de guerra.



As armas de guerra (navios e aeronaves) usados no jogo podem ser de 5 tipos, sendo permitido posicionar no tabuleiro apenas um número pré-determinado de elementos de cada tipo. Ver a tabela na página seguinte.

Tipo	Forma	Quantidade Permitida
Submarino		4
Hidroavião		3
Cruzador		3
Encouraçado		2
Porta-aviões		1

Para colocar uma figura associada a uma arma de guerra no tabuleiro, um jogador informa uma configuração dada por uma posição (onde a referência da figura será colocada (a bola branca)) e uma orientação, que pode ser NORTE, SUL, LESTE, OESTE. Uma figura está na configuração padrão se sua referência tem índices (0,0) e aponta para o oeste, conforme a tabela. Uma configuração é **válida** se:

I. A figura está **totalmente contida no tabuleiro.**

II. A figura **não toca nenhuma outra figura** em qualquer direção.

III. A figura **não intersecta outra figura.**

Por exemplo, o hidroavião na parte superior esquerda da figura na página anterior está na coordenada E4, orientado para a direção LESTE. Para implementar um jogo de Batalha Naval:

a) Descreva os tipos de dados em C necessários para representar os objetos e conceitos do jogo Batalha Naval (2.0 pontos).

b) Implemente uma função em C que receba um par de coordenadas (x,y), $x \in \{0,...15\}$ e $y \in \{A,...,O\}$, e retorne 1, se uma célula ocupada foi atingida e 0 caso contrário.

c) Implemente uma função em C que receba uma arma de guerra e uma configuração e produza uma nova figura de acordo com a configuração desejada.

d) Implemente uma função em C que receba como argumentos um tabuleiro e uma arma de guerra em uma configuração, e retorne 1 se a configuração da figura correspondente é válida e 0 caso contrário.

e) Implemente uma função em C que receba como argumentos um tabuleiro, e uma de arma de guerra com uma configuração válida e marque as células ocupadas.

3) Considere um arquivo texto contendo registros que descrevem círculos, onde cada círculo é definido por um identificador (id), dado por um número inteiro, seu raio (r) em cm, sua posição (x,y) no plano em cm e seu vetor velocidade (vx, vy) em cm/s.

0 3.0 10.0 10.0 10.0 10.0

1 5.0 20.0 20.0 10.0 10.0

2 10.0 50.0 100.0 5.0 5.0

...

a) Descreva a estrutura de dados usada para representar um conjunto de círculos.

b) Implemente uma função que determine se dois círculos estão em colisão.

b) Implemente uma função que permita acionar um elemento ao conjunto.

c) Implemente uma função que leia um conjunto de círculos de um arquivo texto para memória usando a estrutura definida em a).

d) implemente uma função que, supondo que as posições lidas do arquivo são dadas no instante de tempo 0s, atualize a posição de cada círculo no instante de tempo t , passado como parâmetro.

4) Considere uma matriz ***n***x***m*** onde os elementos podem assumir dois valores do tipo char: '*' e '0'. Escreva um programa que:

a) Leia de um arquivo texto o conjunto o número de linhas e colunas da matriz e armazene os valores em uma matriz alocada de forma dinâmico como um array de ponteiros.

b) Gere uma nova matriz de inteiros, também alocada como a matriz de caracteres onde cada elemento armazena o número de bombas vizinhas a posição considerada na matriz de bombas (matriz de caracteres). Considere uma vizinhança 8, isto é, os oito pontos cardeais.

5) Considere strings representadas por uma cadeias de caracteres. Escreva funções em C que efetuem as seguintes funcionalidades:

a) Dada uma string s, retornar sua duplicata.

b) Retornar o número de caracteres de uma string s.

c) Dadas duas strings s e t, verificar se t está contida em s, e retornar sua posição inicial em s. Retornar -1 caso t não seja substring de t.

d) Dadas duas strings s e t, verificar se os caracteres de t formam uma subsequência de s. Exemplo: “rato” e “buraco” são subsequências de “burocrático”, mas “bota” não é.

e) Escreva uma função em C que determine se uma string s é anterior a string t considerando a ordenação lexicográfica.

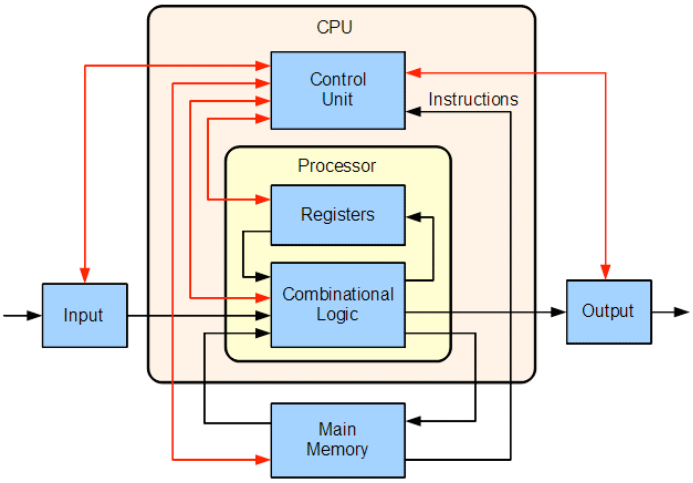
6) Considere uma matriz nxm de números reais que descrevem as alturas de um terreno. Escreva um programa que determine pontos de máximo locais. Um ponto é de máximo local se seu valor é maior do que todos os seus vizinhos dentro de uma distância menor que 2 considerando a métrica dist(p0,p1) = |p0.x-p1.x|+|p0.y-p1.y|.

7) Considere uma matriz de caracteres de tamanho n x m que descreve um jogo de palavras cruzadas. Escreva um programa em C, tal que, dada um string s, verifique se s encontras-se na matriz.

8)Escreva em C funções que que permitam manipular (acessar e modificar elementos) uma matriz tridimensional de dimensões nxmxo através de um armazenamento como array unidimensional.

9) Escreva uma função em C que implemente uma busca ternária (https://en.wikipedia.org/wiki/Ternary_search) em um array.

10) Um computador é composto de vários componentes como CPU, memória principal, dispositivos de entrada e dispositivos de saída. Um processo computacional ocorre através da interação entre os vários componentes do sistema, os dados e o usuário.



Fonte: https://en.wikipedia.org/wiki/Central_processing_unit#/media/File:ABasicComputer.gif

Escreva um programa em C que simule um computador simplificado.

A memória do computador simplificado é caracterizada por um conjunto de palavras de 32bits. A CPU é composta de uma unidade lógica e aritmética capaz de executar um pequeno conjunto de instruções. Além disso, ela possui três registradores: o contador de programa (pc) que armazena o endereço da próxima instrução a ser executada; o contador de instruções (ir)que armazena a instrução corrente e o acumulador (acc) que é um registrador utilizado para dar suporte às operações. As instruções disponíveis encontram-se abaixo, com os respectivos códigos em hexadecimal e suas descrições:

Instrução	Código (hexadecimal)	Explicação	Exemplo de opcode
NVALID	0x0	Instrução invalida	00 00 00 00
READ	0x1	Le uma palavra do dispositivo de entrada (o console) para o acumulador	00 01 00 00
WRITE	0x2	Escreve um valor armazenado no acumulador no dispositivo de saida (o console)	00 02 00 00
LOAD	0x3	Carrega para o acumulador uma palavra da memoria no endereco XX especificado pelos 2 bytes menos significativos	00 03 XX XX
STORE	0x4	Armazena na memoria, no endereco XX, a palavra contida no acumulador	00 04 XX XX
ADD	0x5	Adiciona o valor no endereço de memória XX ao valor no acumulador	00 05 XX XX
SUB	0x6	Subtrai o valor no endereço de memória XX do valor no acumulador	00 06 XX XX
MUL	0x7	Multiplica o valor no acumulador pelo valor no endereço de memória XX	00 07 XX XX
DIV	0x8	Divide o valor no acumulador pelo valor no endereço de memória XX	00 08 XX XX
MOD	0x9	Determina o resto da divisão do valor no acumulador pelo valor no endereço de memória XX	00 09 XX XX
BRANCH	0xA	Transfere o fluxo de execução para o endereço de memória XX	00 0A XX XX
BRANCHNEG	0xB	Transfere o fluxo para o endereço XX caso o valor no acumulador seja negativo	00 0B XX XX
BRANCHZERO	0xC	Transfere o fluxo para o endereço XX caso o valor no	00 0C XX XX

		acumulador seja igual à zero	
HALT	0xD	Aborta o programa	00 0D XX XX

A versão simplificada do computador executa um programa carregado em memória através do seguinte conjunto de passos:

1. A próxima instrução a ser executada é buscada no endereço identificado pelo *contador de programa* e armazenada no *contador de instruções*.
2. O *opcode* de 32 bits no *contador de instruções* é decodificado, sendo extraída a instrução (*operation*) e o operando (*operand*), caso exista.
3. A instrução é executada.
4. O *contador de instruções* é atualizado conforme o fluxo determinado pela instrução executada.

O sistema deve ser subdividido em três módulos: principal (main.c), cpu (cpu.c, cpu.h), memória (mem.c, mem.h). O módulo principal (main.c) utiliza os demais módulos.

No arquivo cabeçalho(mem.h) do modulo *mem* encontramos a definição do tipo **word_t** que descreve uma palavra da memória e do tipo **memory_t**, que descreve

```
#ifndef MEM_H_INCLUDED
#define MEM_H_INCLUDED
#ifdef HUGE
/*Tamanho maximo da memoria.Espaco de enderecamento de  16 bits*/
#define MAX_MEM_SIZE 65536
#else
#define MAX_MEM_SIZE 256
#endif

typedef int word_t;

typedef word_t memory_t[MAX_MEM_SIZE];

/* Limpa todo o conteudo da memoria usada pela CPU */
void clearMem(memory_t mem);

/* Imprime no console todos os enderecos de memoria e seus conteudos */
void dumpMem(const memory_t mem);

/* Retorna uma palavra da memoria no endereco especificado*/
word_t getWord(const memory_t mem, int address);

/* Armazena uma palavra na memoria*/
void setWord(memory_t mem,int address, word_t word);

#endif // MEM_H_INCLUDED
```

um array de word_t. Além disso temos as assinaturas das funções que manipula a memória:

O módulo cpu deve descrever a CPU como um tipo definido pelo usuário baseado em um tipo estruturado (*struct*) da linguagem C. Observe que a CPU conhece (tem uma dependência) para a memória e retém seu endereço em um dos seus campos.

```
#ifndef CPU_H_INCLUDED
#define CPU_H_INCLUDED

#include"mem.h"

#define NUM_INST 14

enum instruction{INVALID = 0x0,  READ = 0x1, WRITE = 0x2, LOAD = 0x3, STORE = 0x4, ADD = 0x5, SUB = 0x6,
            MUL = 0x7,DIV = 0x8, MOD = 0x9, BRANCH = 0xa, BRANCHNEG = 0Xb, BRANCHZERO = 0xc,HALT =
            0xd};

typedef enum instruction instruction_t;

struct decode {char * inst_name;
           word_t opcode;} decode;

struct _cpu
{
   memory_t mem;
   word_t acc;
   int pc;

};

typedef struct _cpu cpu_t;

/* Array contendo pares <nome_instrucao, codigo_instrucao> */
extern struct decode decode_list[NUM_INST];

/* retorna um opcode, isto e, uma palavra de memoria com a instrucao e o operando */
word_t getOpcodeCPU(const cpu_t cpu);

/* Inicia o funcionamento da CPU, zera o contador de programa e limpa a memoria */
void init(cpu_t *cpu);

/* Roda um programa previamente carregado na memoria da CPU*/
void run(cpu_t *cpu);

void exec(cpu_t *cpu);

/* Sai da cpu */
void exitCPU(const cpu_t cpu);

/* Mostra o conteudo da CPU */
void dumpCPU(const cpu_t  cpu);

/* Le um arquivo com um programa em linguagem de montagem e compila gerando um codigo objeto*/
void assembleProgram(char *src_file_name, char *obj_file_name);

/* Carrega em memoria os bytes de um programa executável*/
void loadProgram(cpu_t *cpu, char *file_name);

#endif // CPU_H_INCLUDED
```

Escreva um programa em C que seja capaz de ler um código na linguagem de montagem “Tinyasm” com extensão .ty e monte os bytecodes gerando o código objeto, que deve ser carregado em memória, antes da execução. Cada *opcode* traduzido é carregado em memória em ordem crescente de endereço.

Exemplo de programa na linguagem de máquina do computador simplificado:

```
LOAD 7 ; carrega o valor no endereço 7 no acumulador
WRITE  ; escreve o valor do acumulador
LOAD 8 ; carrega o valor no endereço 8 no acumulador
WRITE ; escreve o valor do acumulador
ADD 7 ; adiciona o valor no endereço 7 no acumulador
WRITE ; escreve o valor do acumulador
HALT ; interrompe a execução
4
5
```

```
LOAD 11 ; acc = n
BRANCHZERO 10 ; se (n=0) salte pare
LOAD 12 ; acc = sum
ADD 13 ;  acc = acc + inc
STORE 12 ; sum = sum + inc
WRITE ; escreve sum
LOAD 11 ; acc = n
SUB 13 ; acc = acc - inc
STORE 11 ; n = n - 1
BRANCH 0 ; retorn a primeira instrucao
HALT ; termina o programa
3 ; (n = 3)
0 ; (sum = 0)
1 ; (inc = 1)
```