PROJETO FINAL DE SISTEMAS OPERACIONAIS I

Implementação do escalonador um mini-SO em assembly

Caracteristicas:

- Programação em modo real
- Multitarefa (escalonamento de processos)

Memória:

- Limitada a 1Mb (modo real)
- Gerenciamento por partição fixa (32k por partição)
- Cada partição está reservada para um único processo
- Mapa de memória

	Offset		
Segmento	***Início***	***Fim***	***Uso***
0000h	0000h	03FFh	Vetores de Interrupção
	0400h	05FFh	Dados da BIOS/Sistema
	0600h	7BFFh	Dados do SO
	7C00h	7DFFh	BootLoader
	7E00h	7FFFh	não usado
0800h	0000h	7FFFh	Sistema Operacional - kernel
1000h	0000h	7FFFh	Partição 0 de memória
1800h	0000h	7FFFh	Partição 1 de memória
2000h	0000h	7FFFh	Partição 2 de memória
2800h	0000h	7FFFh	Partição 3 de memória
...	...	...	...
9000h	0000h	7FFFh	Partição 16 de memória
9800h	0000h	7FFFh	Partição 17 de memória
>A000h			Entrada/Saída

TAREFAS:

1) Fazer funções de conversão de inteiros para string, permitindo a impressão de valores:
   a) Converter inteiro tipo char (1 byte) para string hexadecimal
   b) Converter inteiro tipo short (2 bytes) para string hexadecimal
   c) Converter inteiro tipo char (1 byte) para string decimal
   d) Converter inteiro tipo short (2 bytes) para string decimal
2) Fazer as funções de manipulação de strings:
   a) Copiar uma string para outra
   b) Concatenar duas strings
   c) Encontrar o número de caracteres de uma string

3) Definir a estrutura contendo as informações sobre cada processo :
    - Cada entrada precisa armazenar (pelo menos)
        - Estado: Pronto=0, Não usado=1, Bloqueado=2, Rodando=3
        - Registradores de Pilha: SS e SP
        - Tempo de execução – Número de vezes que o processos rodou
4) Definir as variáveis com informações sobre os processos:
    a) Variável contendo o processo rodando
    b) Tabela de processos
    - A tabela de processos deve ter 19 entradas (1 entrada para o sistema operacional)
5) Fazer Rotina de inicialização da tabela de processos
    a) Marcar processo 0 (kernel) como rodando
    b) Marcar demais processos como não usado
6) Fazer função que busca processo não usado na tabela, retornando o índice (1 byte). Se não encontrar, retornar -1=0FFh
7) Fazer função que dado um índice de 0 à 19, retorna o segmento do mesmo (2 bytes).
8) (Assembly) Fazer função que dado o endereço do código (segmento e deslocamento), o endereço de destino e o número de bytes, copie o código para o destino.
9) (Assembly) Fazer função que dado o índice do processo, inicialize a pilha do processo (FLAGS, CS, IP, registradores) e o topo da pilha na entrada correspondente da tabela de processos.
10) Fazer função que dado um endereço de código, torne-o um processo:
    a) Busque na tabela um processo não usado (chame a função da tarefa tarefa 6)
    b) Encontre o segmento de memória correspondente (chame a função da tarefa tarefa 7)
    c) Carregue o código no segmento encontrado (chame a função da tarefa tarefa 8)
    d) Inicialize a pilha do processo (chame a função da tarefa 9)
    e) Coloque o processo no estado pronto
11) Fazer função que busque o próximo processo no estado pronto (a partir do que está rodando), retornando o índice.
12) (Assembly) Faça o código do novo manipulador da interrupção 08h:
    a) Salve demais registradores (AX, BX, CX, DX, DS, ES, BP, DI, SI) na pilha
    b) Passe o processo atual para o estado pronto
    c) Armazene SS e SP na tabela de processos
    d) Carregue o SS e SP do processo do SO
    e) Chame a interrupção original do relógio
    f) Encontre o próximo processo no estado pronto (tarefa 6)
    g) Armazene o SS e SP do processo do SO
    h) Carregue SS e SP do novo processo
    i) Recupere os registradores
13) (Assembly) Fazer a função que troca o vetor da interrupção 08h pelo endereço do manipulador da tarefa 12.
14) Implementar o kernel:
    a) Chame a função de inicialização da tabela de processos (tarefa 5)
    b) Chame a função que troca o vetor de interrupção do relógio pela nova interrupção (tarefa 13)
    c) Carregue os programas exemplos como processos diferentes (tarefa 10)