Computer Engineering

Computer Science

Interrupção em microprocessadores

Felipe Gabriel
Alison Bruno
Vítor Gabriel

Objetivo

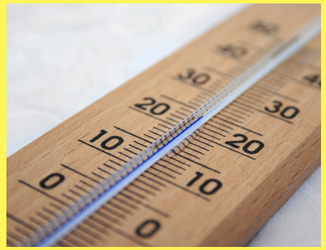
Interromper um processo, caso uma ação prioritária seja acionada. Exemplo: sensores de uma máquina industrial controlada por um micro controlador.

Interrupções usadas

Botão

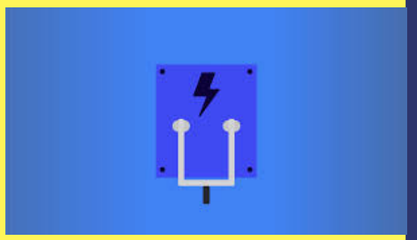
Prioridade 1

Temperatura



Prioridade 2

Energia

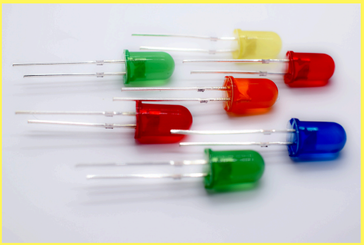


Prioridade 3

Manutenção

Prioridade 4

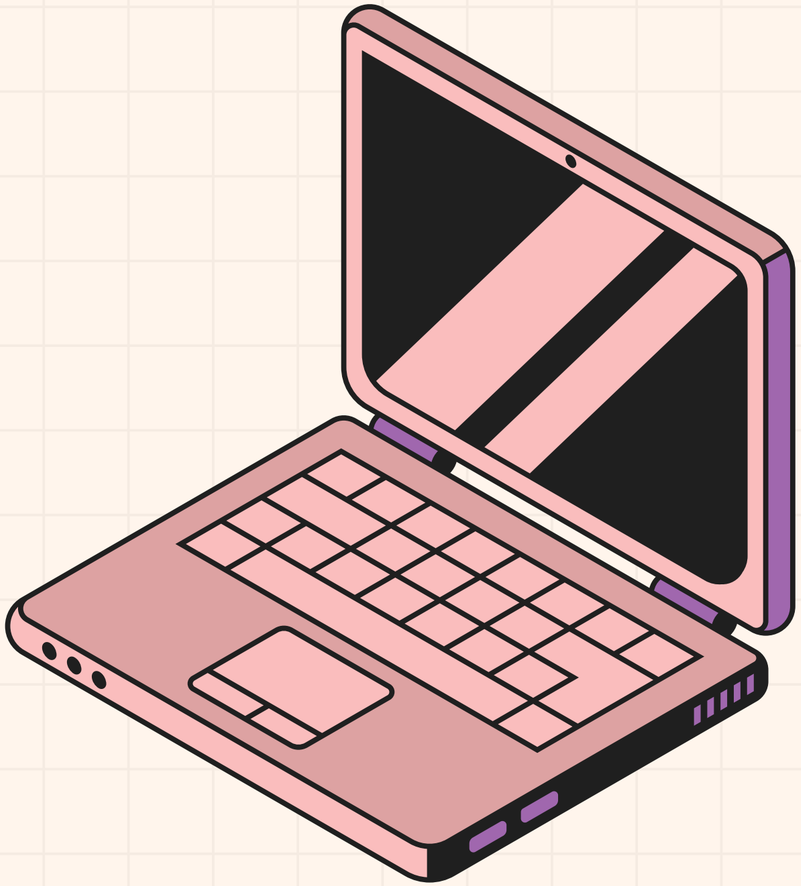
LED



Prioridade 5

RESOLUÇÃO DO PROBLEMA: FILA DE PRIORIDADE

- Definimos cada interrupção como uma prioridade, indo da mais importante (1) até a menos importante (5);
- O microprocessador irá executar as interrupções de acordo com a prioridade de cada uma;



Funções de interrupção

- ISR é uma função executada, pelo microcontrolador, quando uma interrupção é chamada

```c
// ---------- Simulando as Interrupções (ISRs) ----------
void ISR_botao_emergencia(Fila* fila) {
    printf("[ISR] Botão de Emergência acionado!\n");
    adicionar_evento(fila, criar_evento("Botão de Emergência", 1));
}

void ISR_temperatura_critica(Fila* fila) {
    printf("[ISR] Temperatura Crítica detectada!\n");
    adicionar_evento(fila, criar_evento("Temperatura Crítica", 2));
}

void ISR_queda_energia(Fila* fila) {
    printf("[ISR] Queda de Energia detectada!\n");
    adicionar_evento(fila, criar_evento("Queda de Energia", 2));
}

void ISR_manutencao_preventiva(Fila* fila) {
    printf("[ISR] Manutenção Preventiva necessária.\n");
    adicionar_evento(fila, criar_evento("Problemas de Manutenção", 3));
}

void ISR_LedApagando(Fila* fila) {
    printf("[ISR] Sistema funcionando normalmente.\n");
    adicionar_evento(fila, criar_evento("Led apagando", 5));
}
```

```c
void tratar_eventos(Fila* fila) {
    Evento* atual = NULL;

    //Enquanto a fila tiver elementos
    while (fila->inicio != NULL) {
        atual = remover_evento(fila); // Pega o elemento de maior prioridade

        printf("\n>>> Iniciando tratamento: %s (Prioridade %d)\n", atual->descricao, atual->prioridade);

        for (int i = 0; i < 3; i++) {
            printf("... Trabalhando no evento: %s [Passo %d/3]\n", atual->descricao, i+1);

            // Perguntar se o usuário quer simular uma nova interrupção
            //So me pergunta uma vez para fins praticos de demonstracao
            if( i == 0){

                printf("Deseja simular uma nova interrupção? (s/n): ");
                char resposta;
                scanf(" %c", &resposta);
                getchar(); // limpar '\n'

                if (resposta == 's' || resposta == 'S') {
                    simular_ISR_durante_tratamento(fila);

                    // Checa se há um evento de prioridade maior
                    if (fila->inicio && fila->inicio->prioridade < atual->prioridade) {
                        printf("\n!!! Preempção detectada !!!\n");
                        printf("Novo evento mais urgente: %s (Prioridade %d)\n", fila->inicio->descricao,
fila->inicio->prioridade);

                        // Guarda o evento atual de volta na fila
                        adicionar_evento(fila, atual);
                        break; // sai do for para tratar o mais urgente
                    }
                }
            }
        }

        // Se o evento foi totalmente tratado (não preemptado)
        if (fila->inicio == NULL || atual->prioridade <= fila->inicio->prioridade) {
            printf(">>> Evento tratado: %s\n", atual->descricao);
            free(atual);
        }
    }

    printf("\nTodos os eventos foram tratados.\n");
}
```

Preempção: Quando um evento mais urgente é chamado, enquanto um evento está sendo tratado/executado

```c
int main() {
    Fila* fila_eventos = criar_fila();
    int opcao;

    do {
        menu();
        printf("\nEscolha uma opção: ");
        scanf("%d", &opcao);
        getchar(); // limpar o '\n'

        switch (opcao) {
            case 1:
                ISR_botao_emergencia(fila_eventos);
                break;
            case 2:
                ISR_temperatura_critica(fila_eventos);
                break;
            case 3:
                ISR_queda_energia(fila_eventos);
                break;
            case 4:
                ISR_manutencao_preventiva(fila_eventos);
                break;
            case 5:
                ISR_LedApagando(fila_eventos);
                break;
            case 6:
                tratar_eventos(fila_eventos);
                break;
            case 7:
                imprimir_eventos(fila_eventos);
                break;
            case 0:
                printf("Finalizando o sistema...\n");
```

Preempção: Quando um evento mais urgente é chamado, enquanto um evento está sendo tratado/executado

EXECUÇÃO DO CÓDIGO