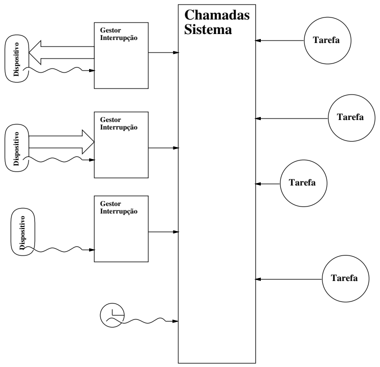
Sumário

- Executivos multitarefa.

  RMX/E286.
- Estrutura do núcleo.
- Escalonamento.
- Interrupções.

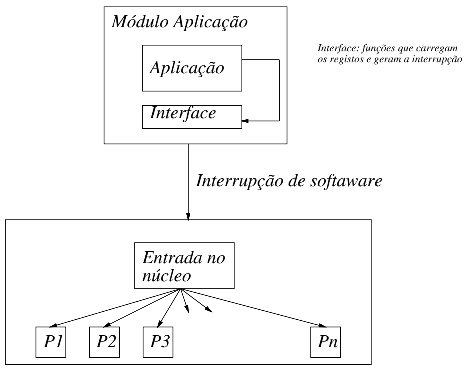
Arquitectura



Estrutura do Núcleo do RMX/E286

- Gestão das interrupções.
- Multiplexagem do processador.
- Gestão de Processos (Tarefas).
- Semáforos e Regiões.
- Gestão de Memória.
- Caixas de Correio.

Interface ao núcleo



Funções do núcleo

- Comutação de tarefas.
- Controlo de acesso aos recursos do sistema.
- Comunicação entre tarefas.
- Resposta do sistema a acontecimentos externos.

Objectos do núcleo

- Tarefas.
- Segmentos.
- Semáforos.
- Regiões.
- Caixas do correio.
- Filas de mensagens.

Manipulação dos objectos

- Os objectos são mantidos em estruturas internas ao núcleo.

  Listas de descritores.

- Os objectos são conhecidos pela aplicação através de identificadores.

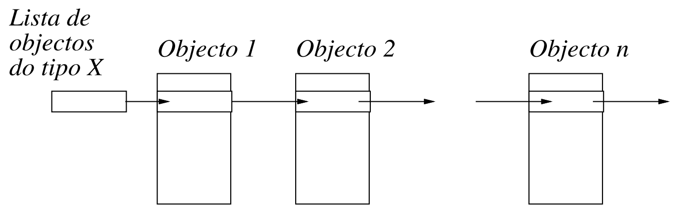
  Designados por chaves ("tokens").

- A memória para as estruturas do núcleo é reservada estdaáticamente.

  Configuração do núcleo.

Estruturas de dados

Lista ligada de objectos (uma para cada tipo)



Criação de objectos

- Primitiva do tipo `RQ$CREATE$TIPODOOBJECTO`.
- Retira da lista correspondente de objectos livres um objecto.
- Inicia o descritor do objecto.
- Insere na lista de objectos em utilização.
- Devolve uma chave à aplicação.

  Pode ser um inteiro (ex: índice numa tabela).

  Pode ser um endereço.

Eliminação de objectos

- Primitiva do tipo `RQ$DELETE$TIPODOOBJECTO`.
- Actualiza estruturas de dados associadas a esse objecto.

  Ex: deve libertar registo do objecto.
- Retira o objecto da lista de objectos utilizados.
- Insere o objecto na lista de objectos livres.

Tarefas

- Objectos activos do núcleo.
- Programas sequenciais.
- Carregamento em memória.

  Deve ser feito previamente (o sistema não suporta o acesso a memória secundária).
- `CREATE$TASK (prioridade, procedimento, segmentoDados, segmentoPilha, tamanhoPilha, flags, apontCodigoRetorno)`

Escalonamento

- Baseado em prioridades

  0 (mais alta) até 255 (mais baixa).
- Premptível.

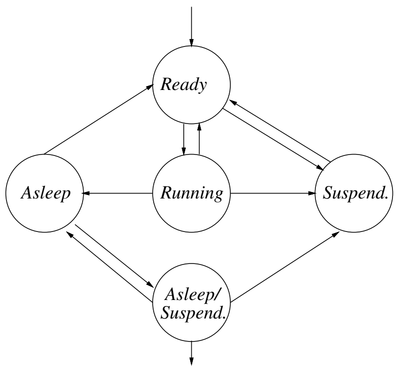
  Não é utilizado "quantum" de tempo.
- Uma tarefa executa-se até:

  Terminar.

  Bloquear-se.

  Uma tarefa mais prioritária fique executável.

Transições de estado de tarefas



Regras de escalonamento

- Uma tarefa quando criada fica `Ready`.
- Define-se a tarefa candidata como sendo a tarefa no estado `Ready` com maior prioridade.

  Ou, quando existe mais do que uma tarefa com essa prioridade, a que está à mais tempo à espera do processador.

- Uma tarefa candidata fica `Running` quando:

  Tem maior prioridade que a tarefa `Running` *ou quando a tarefa* `Running` *abandona voluntariamente o processador.*

- Uma tarefa `Running` fica `Ready` quando é retirada do processador por uma tarefa de maior prioridade.

Regras de escalonamento (continuação)

- Uma tarefa `Running` fica `Asleep` quando:

  Se coloca nesse estado voluntariamente.

  Requisita um recurso não disponível.

- Uma tarefa passa de `Asleep` para `Ready` ou de `Asleep-Suspended` para `Suspended` quando:

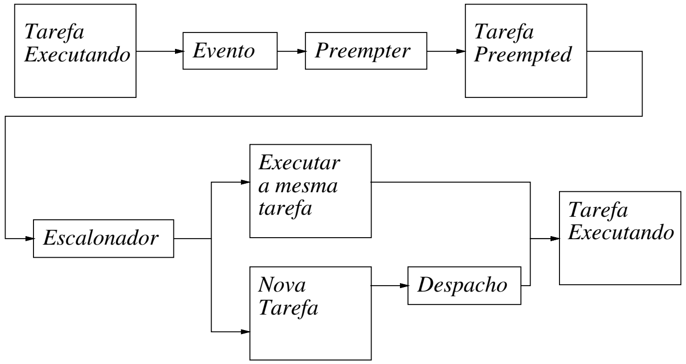
  O pedido que a fez esperar é atendido.

  Acaba o tempo de espera.

Regras de escalonamento (continuação)

- Uma tarefa passa de `Running` para `Suspended` quando se suspende a si própria.
- Uma tarefa passa de `Ready` para `Suspended` ou de `Asleep` para `Asleep-Suspended` quando é suspensa por outra tarefa.
- Uma tarefa passa de `Suspended` para `Ready` ou de `Asleep-Suspended` para `Asleep` quando é instruída para prosseguir.

Comutação



Interrupções

- Gestores de interrupção ("handler"):

  Rotinas invocadas quando existe uma interrupção.

- Tarefas:

  Que ficam à espera da interrupção.

- `SET$INTERRUPT (nivel, pendentes, handler, segDados, &erro)`

  Especifica o número máximo de tarefas pendentes nesse nivel.

Interrupções (continuação)

- `ENTER$INTERRUPT (nivel, &erro)`

  Muda o segmento de dados para o segmento da tarefa associada.

- `EXIT$INTERRUPT (nivel, &erro)`

  Volta a permitir a sinalização de interrupções.

Tarefa de interrupção

- Para evitar ter as interrupções inibidas por um período demasiado longo:

  Decompõe-se o processamento em duas fases: (i) processamento imediato a realizar pelo "handler" e; (ii) processamento diferido a realizar pela tarefa.

- `WAIT$INTERRUPT (nivel, &erro)`

  Coloca a tarefa em espera.

- `SIGNAL$INTERRUPT (nivel, &erro)`

  Sinaliza a tarefa.

Interrupções exemplo (handler)

```
handlerInt ()
{
  /* salvaguardar os registos */
  enterInterrupt (nivel, &erro);
  if (processar_ja) {
    /* Processar */
    exitInterrupt (nivel, &erro);
  }
  else
    signalInterrupt (nivel, &erro);
  /* salvaguardar os registos */
}
```


Interrupções exemplo (tarefa)

```
tarefaInt ()
{
  setInterrupt (nivel, 1, &handlerInt, seg_dados, &erro);
  /* Iniciacoes */
  while (1) {
    waitInterrupt (nivel, &erro);
    /* tratamento diferido */
  }
}
```


Resumo

- Estrutura do núcleo:

  Arquitectura.

  Objectos.

  Interface.
- Gestão de tarefas.
- Gestão das interrupções.