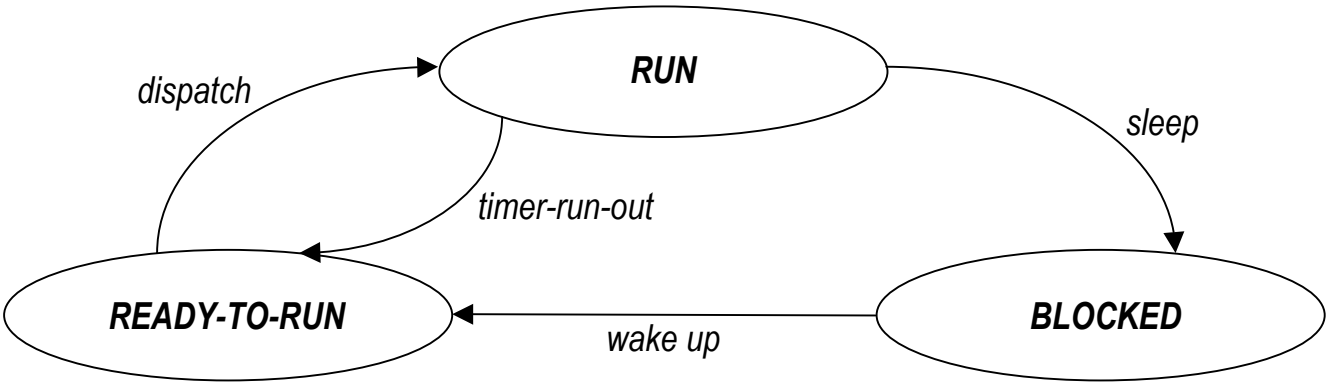
Parte A (10 valores)

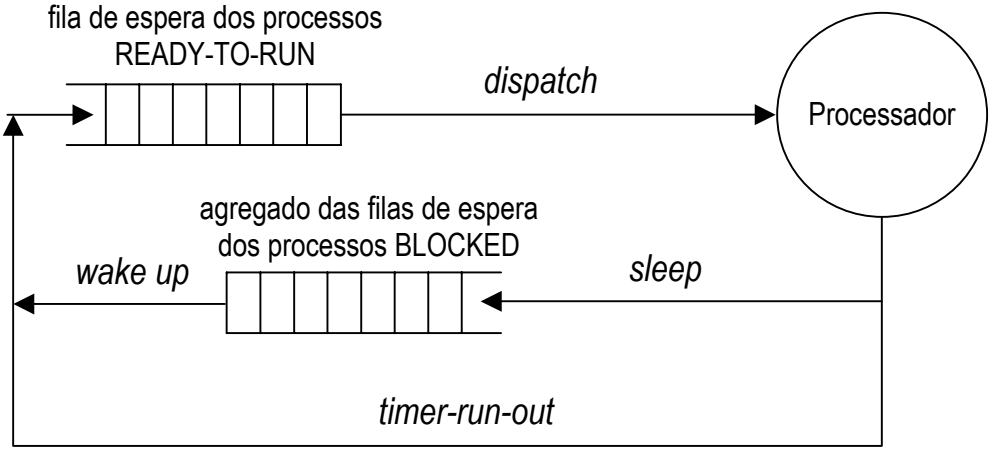
1. Será possível conceber-se multiprocessamento sem multiprogramação? Em que circunstâncias?
2. Que critérios devem ser satisfeitos pelos algoritmos de 'scheduling' do processador? Quais são os mais importantes num sistemas multi-utilizador de uso geral?
3. Qual é a diferença entre uma organização de memória virtual segmentada de uma paginada? Quais são as vantagens e inconvenientes de cada uma delas?
4. Porque é que sobre a abstracção da memória de massa, concebida como um 'array' de blocos para armazenamento de dados, se torna fundamental introduzir o conceito de sistema de ficheiros? Quais são os elementos essenciais que definem a sua arquitectura.?
5. Porque é que num ambiente multiprogramado é fundamental desacoplar a comunicação dos processos utilizadores com os diferentes dispositivos de entrada-saída?

Parte B (10 valores)

A figura apresenta o diagrama de transição de estados do 'scheduling' de baixo nível do processador.



Suponha que foi implementada uma disciplina de 'scheduling' de tipo 'round-robin', como a figura abaixo ilustra.



Admita ainda que foram definidas as estruturas de dados seguintes:

Entrada (simplificada) da Tabela de Controlo de Processos

```
typedef struct
        { BOOLEAN busy;                /* sinalização de entrada ocupada */
          unsigned int pid,                 /* identificador do processo */
                       pstat;             /* estado do processo: 0 - RUN;
                                             1 - BLOCKED; 2 - READY-TO-RUN */
          unsigned char intreg[K];             /* contexto do processador */
          unsigned long addspace;       /* endereço da região de memória
                 principal onde está localizado o espaço de endereçagem
                                       do processo (org. de memória real) */
        } PCT_ENTRY;
```

Nó de lista biligada

```
struct binode
      { unsigned int info;                              /* valor armazenado */
        struct binode *ant,                  /* ponteiro para o nó anterior */
                      *next;                 /* ponteiro para o nó seguinte */
      };
typedef struct binode BINODE;
```

FIFO

```
typedef struct
        { BINODE *pin_val,          /* ponteiro para o ponto de inserção */
                 *pout_val;         /* ponteiro para o ponto de retirada */
        } FIFO;
```

Semáforo

```
typedef struct
        { unsigned int val;                         /* valor de contagem */
          FIFO queue;         /* fila de espera dos processos bloqueados */
        } SEMAPHORE;
```

e as variáveis globais descritas abaixo:

```
static SEMAPHORE sem[200];                      /* 'array' de semáforos */
static PCT_ENTRY pct[100];            /* tabela de controlo de processos */
static FIFO redtorun;    /* fila de espera dos processos prontos a serem
                                                          executados */
static unsigned int pindex;              /* índice da entrada da PCT que
                             descreve o processo que detém o processador */
```

Finalmente, as primitivas seguintes estão também disponíveis:

Activação e inibição das interrupções

```
void interrupt_enable (void);
void interrupt_disable (void);
```

Salvaguarda e restauro do contexto

```
void save_context (unsigned int pct_index);
void restore_context (unsigned int pct_index);
```

Reserva e libertação de espaço em memória dinâmica

```
void *malloc (unsigned int size);
void free (void *pnt);
```

Inserção e retirada de nós na FIFO

```
void fifo_in (FIFO *fifo, BINODE *val);
void fifo_out (FIFO *fifo, BINODE *val);
```

Transição de estado dos processos

```
void dispatch (void);
void timer_run_out (void);
void sleep (unsigned int sem_index);
void wakeup (unsigned int sem_index);
```

1. Que tipo de valor deve ser armazenado no campo *info* da FIFO que implementa a fila de espera dos processos prontos a serem executados?

2. Que alterações teriam que ser introduzidas nas estruturas de dados e nas variáveis internas, se se implementasse alternativamente uma disciplina de 'scheduling' de prioridade estática em 10 níveis? Justifique adequadamente a sua resposta.

3. Construa a primitiva que faz o 'down' de um semáforo.

```
void sem_down (unsigned int sem_index);
```

4. Construa a primitiva 'dispatch'.