

Universidade Federal de Roraima
Departamento de Ciência de Computação

DISCIPLINA: Sistemas Operacionais – DCC403

ALUNO(A): Ewelly Fabiane Cunha de Sousa

NOTA: _____

[Questão-1] Utilizando o simulador SOSim (disponível em <http://www.training.com.br/sosim>) apresente os resultado do simulador e uma análise para cada item abaixo.

(PRÁTICA - A)

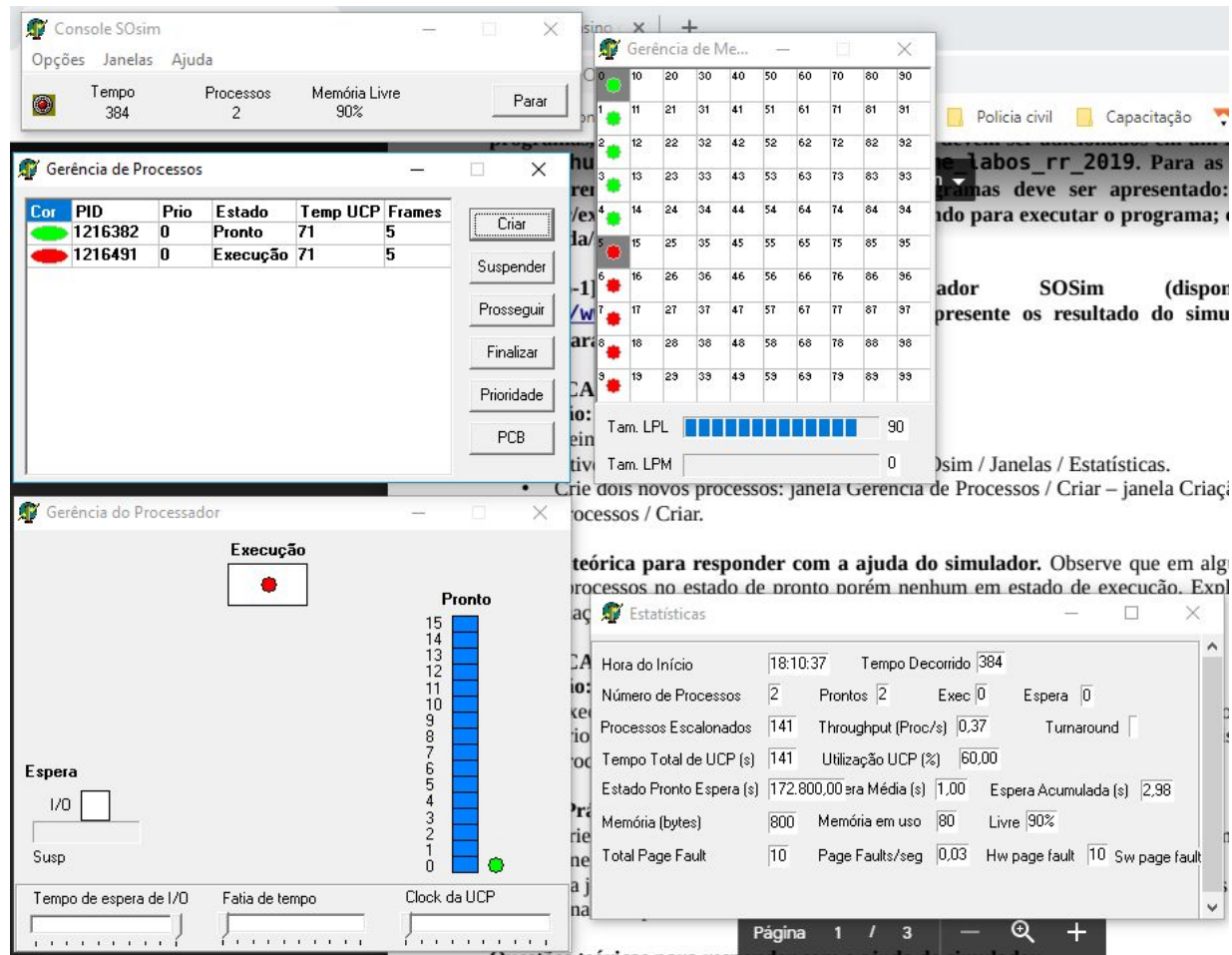
The screenshot displays the SOSim simulator interface with several windows open:

- Console SOSim:** Shows system status: Tempo 134, Processos 2, Memória Livre 90%. Buttons: Opções, Janelas, Ajuda, Parar.
- Gerência de Processos:** A table showing process details:

Cor	PID	Prio	Estado	Temp UCP	Frames
Verde	1216382	0	Execução	9	5
Vermelho	1216491	0	Pronto	8	5

Buttons: Criar, Suspendir, Prosseguir, Finalizar, Prioridade, PCB.
- Gerência do Processador:** Shows the execution state with a bar chart for 'Pronto' (Ready) and 'Execução' (Execution) states. Includes sections for 'Espera' (Waiting) and 'Susp' (Suspended).
- Gerência de Memória:** A grid showing memory allocation status for various processes.
- Estatísticas:** A summary of system performance:

Hora do Início		Tempo Decorrido	
Número de Processos	2	Prontos	2
Processos Escalonados	16	Exec	0
Tempo Total de UCP (s)	16	Espera	0
Estado Pronto Espera (s)	172.800,00	Throughput (Proc/s)	0,12
Memória (bytes)	800	Utilização UCP (%)	50,00
Total Page Fault	10	Espera Média (s)	1,00
		Espera Acumulada (s)	2,81
		Memória em uso	80
		Livre	90%
		Page Faults/seg	0,07
		Hw page fault	10
		Sw page fault	



[Questão teórica] Observe que em alguns momentos existem processos no estado de pronto porém nenhum em estado de execução. Explique o porquê dessa situação.

Essa situação ocorre porque nesse momento os processos estão sendo escalonados por escalonamento circular, ou seja, conteúdo dos registradores do processo em execução são salvos e logo após carrega-se o conteúdo dos registradores do próximo processo na fila. E esse tipo de escalonamento é organizado de uma maneira que cada um deles possua um determinado tempo da CPU e caso um desses processos não termine dentro do seu tempo ele é colocado no fim da fila e outro tempo é dado para o processo no começo da fila. Nas imagens podemos ver o processo pronto na fila esperando para executar e o mesmo executando quando o outro deixa livre o recurso.

(PRÁTICA - B):

The screenshot displays the Console SOsim interface with three main windows:

- Console SOsim:** Shows system status: Tempo 71, Processos 2, Memória Livre 97%, and a Parar button.
- Gerência de Processos:** A table listing processes and their states.
- Gerência do Processador:** A window showing the execution and ready queues.
- Gerência de Me...:** A window showing a memory layout grid and timing information.

Gerência de Processos Table:

Cor	PID	Prio	Estado	Temp UCP	Frames
Verde	2116334	4	Execução	16	5
Vermelho	2129947	3	Pronto	0	5

Gerência do Processador:

- Execução:** Contains a green circle icon.
- Pronto:** A vertical queue of 16 slots (0-15). Slot 3 contains a red circle icon.
- Espera:** Includes I/O and Suspended (Susp) checkboxes.
- Tempo de espera de I/O:** A slider control.
- Fatias de tempo:** A slider control.
- Clock da UCP:** A slider control.

Gerência de Me...:

- Grid:** A 10x10 grid of cells. Cells (0,0), (1,0), and (2,1) contain green circles. Cell (2,2) contains a red circle.
- Tam. LPL:** A bar chart showing 15 blue segments, with a value of 97.
- Tam. LPM:** A bar chart showing 0 segments, with a value of 0.

Console SOsim

Opções Janelas Ajuda

Tempo 110 Processos 2 Memória Livre 96% Parar

Gerência de Processos

Cor	PID	Prio	Estado	Temp UCP	Frames
●	2116334	4	PFault	24	5
●	2129947	3	I/O	3	5

Criar Suspende Prosseguir Finalizar Prioridade PCB

Gerência do Processador

Execução

Pronto

Espera

I/O ● ●

Susp

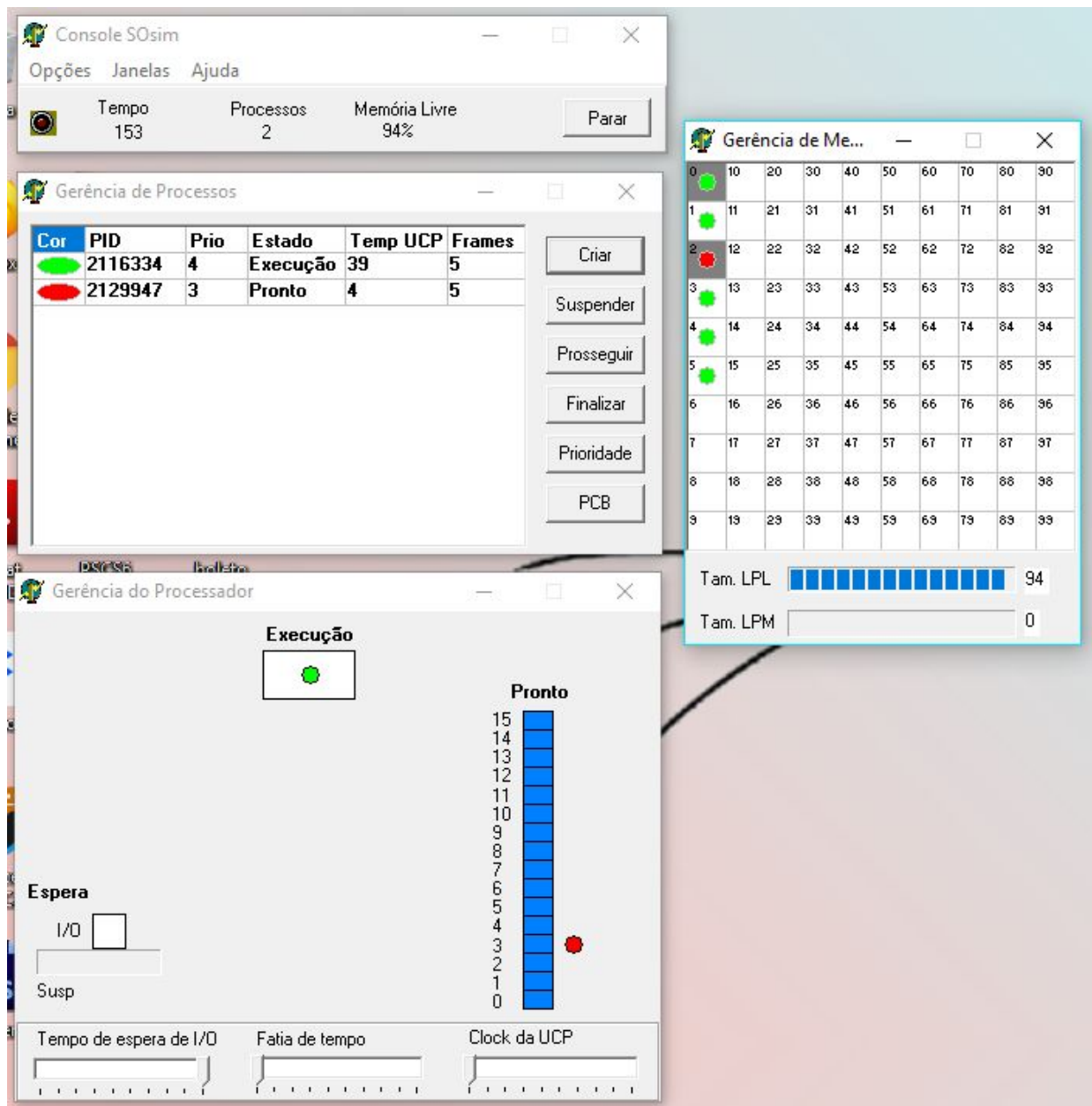
Tempo de espera de I/O Fatia de tempo Clock da UCP

Gerência de Me...

●	10	20	30	40	50	60	70	80	90
●	11	21	31	41	51	61	71	81	91
●	12	22	32	42	52	62	72	82	92
●	13	23	33	43	53	63	73	83	93
●	14	24	34	44	54	64	74	84	94
●	15	25	35	45	55	65	75	85	95
●	16	26	36	46	56	66	76	86	96
●	17	27	37	47	57	67	77	87	97
●	18	28	38	48	58	68	78	88	98
●	19	29	39	49	59	69	79	89	99

Tam. LPL 96

Tam. LPM 0



Questões teóricas para responder com a ajuda do simulador

- Por que o problema do starvation pode ocorrer?

Ocorre porque foi predefinida uma prioridade de nível 4 para o processo de tipo cpu-bound e uma prioridade estática de 3 para o processo I/O-bound. O que acarreta em deixar o processo de menor prioridade em espera eterna até “morrer de fome”.

- Cite duas ações que o administrador do sistema pode realizar quando é identificada a situação de starvation em um processo?

[0] Ignorar o problema kkkkkkk brinks

[1] Usando fila de prioridade onde os dois processos possuem a mesma prioridade ou finalizando ou suspendendo o processo que está causando starvation.

[2] Criar uma fila do tipo FIFO (first in first out) os elementos vão sendo colocados na fila e retirados (processados) por ordem de chegada. A ideia fundamental da fila é que só podemos inserir um novo elemento no final da fila e só podemos retirar o elemento do início.

(PRÁTICA – C) Simulação:

Console SOsim

Opções Janelas Ajuda

Tempo

1032

Processos

2

Memória Livre

90%

Parar

Gerência de Processos

Cor	PID	Prio	Estado	Temp UCP	Frames
<div></div>	4824883	0	Pronto	209	5
<div></div>	4824997	0	Execução	204	5

Criar

Suspender

Prosseguir

Finalizar

Prioridade

PCB

Gerência do Processador

Execução

Pronto

15

14

13

12

11

10

9

8

7

6

5

4

3

2

1

0

Espera

I/O

Susp

Tempo de espera de I/O

Fatias de tempo

Clock da UCP

Gerência de Me...

<div></div>	10	20	30	40	50	60	70	80	90
<div></div>	11	21	31	41	51	61	71	81	91
<div></div>	12	22	32	42	52	62	72	82	92
<div></div>	13	23	33	43	53	63	73	83	93
<div></div>	14	24	34	44	54	64	74	84	94
<div></div>	15	25	35	45	55	65	75	85	95
<div></div>	16	26	36	46	56	66	76	86	96
<div></div>	17	27	37	47	57	67	77	87	97
<div></div>	18	28	38	48	58	68	78	88	98
<div></div>	19	29	39	49	59	69	79	89	99

Tam. LPL

90

Tam. LPM

0

Contexto do Processo

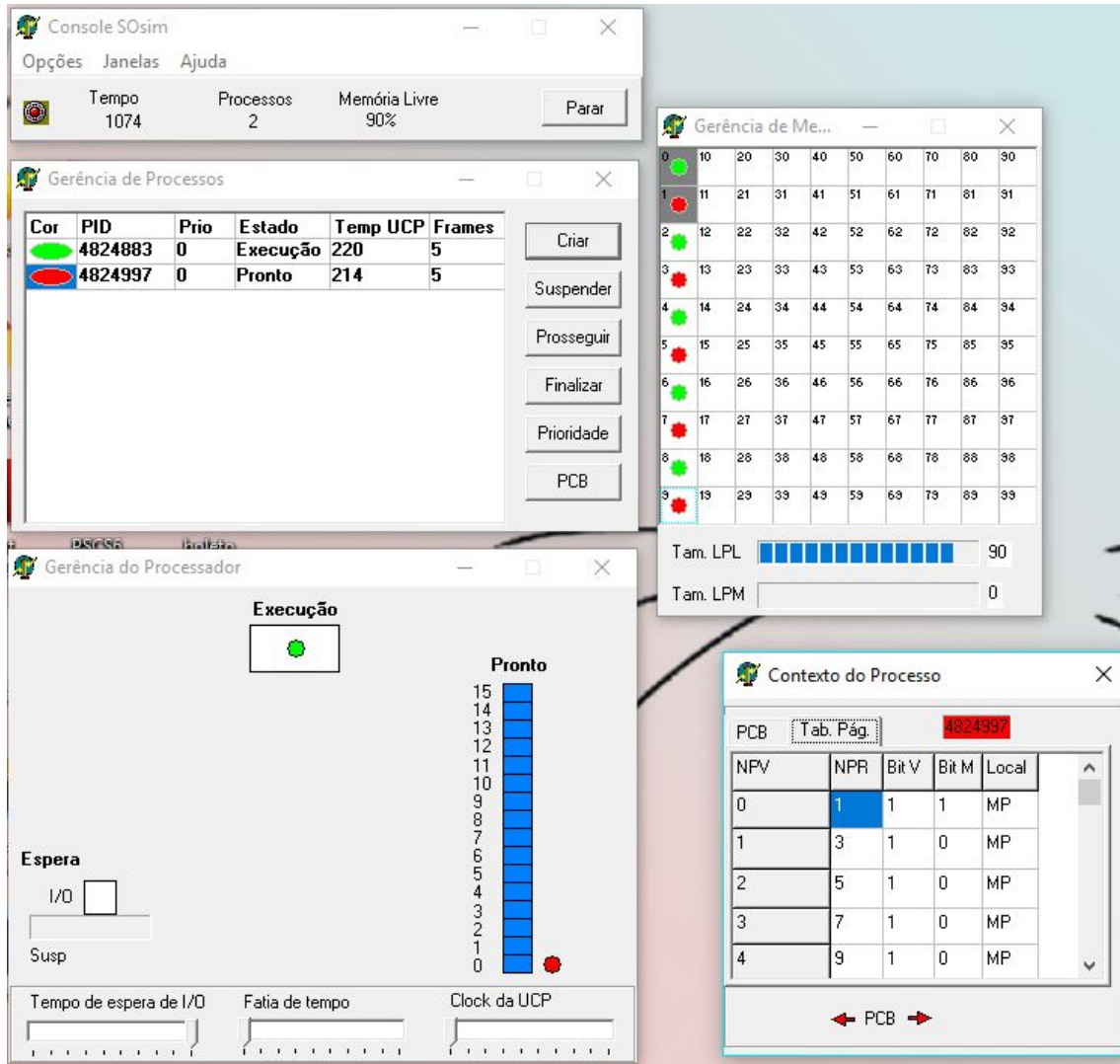
PCB

Tab. Pág

4824883

NPV	NPR	Bit V	Bit M	Local
0	0	1	1	MP
1	2	1	0	MP
2	4	1	0	MP
3	6	1	0	MP
4	8	1	0	MP

← PCB →



Questões teóricas para responder com a ajuda do simulador:

- Qual o espaço de endereçamento real máximo de um processo?

A quantidade total da memória principal e secundária juntas.

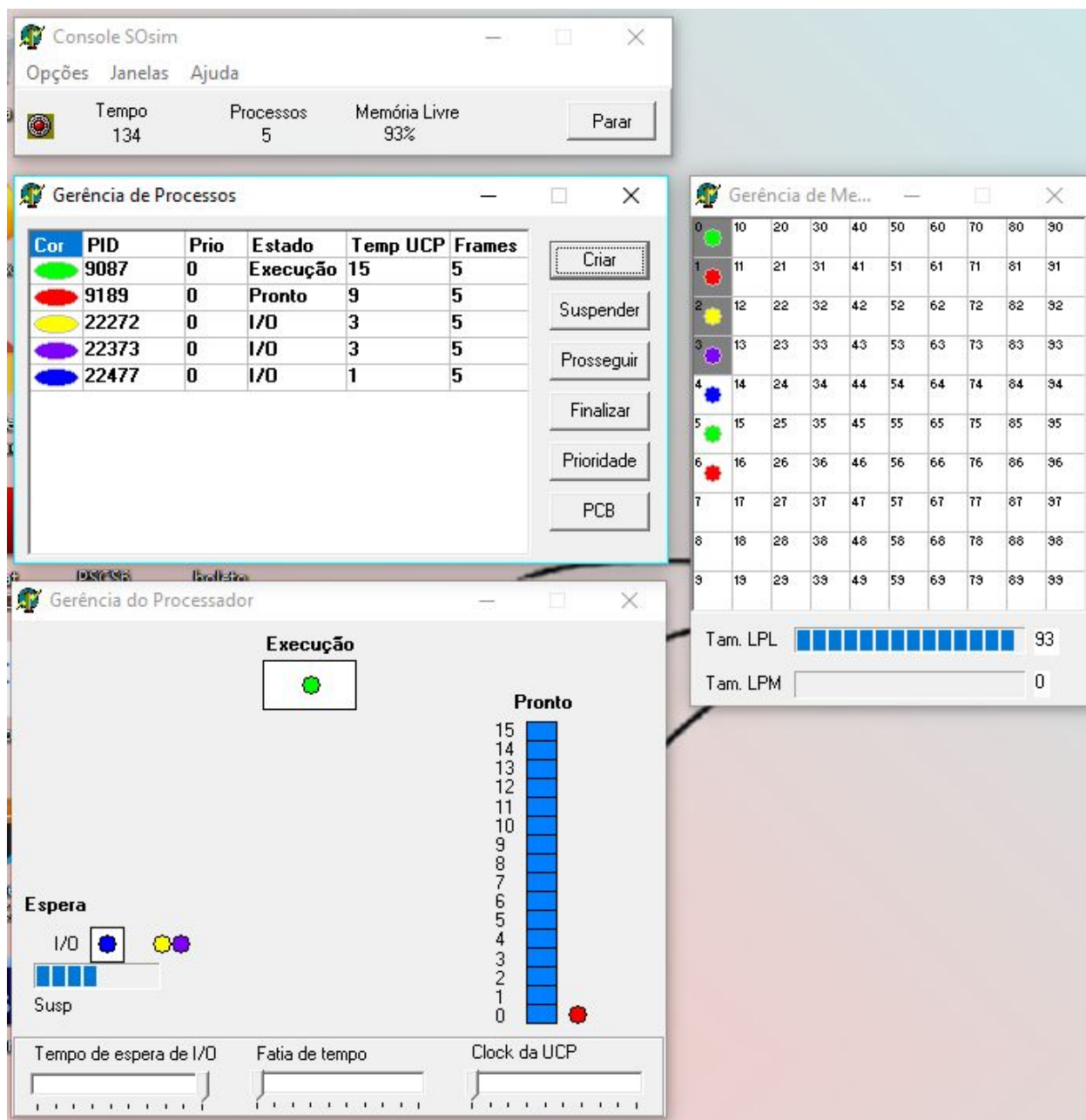
- Qual o espaço de endereçamento real mínimo de um processo?

O tamanho mínimo da tabela de mapeamento carregada.

- Qual o tamanho da página virtual?

O tamanho pode variar de acordo com o processador utilizado e a arquitetura do hardware, podendo em algumas arquiteturas ser configurado.

(PRÁTICA – D)



Console SOsim

Opções Janelas Ajuda

Tempo 196 Processos 5 Memória Livre 92% Parar

Gerência de Processos

Cor	PID	Prio	Estado	Temp UCP	Frames
9087	0	PFault	24	5	
9189	0	PFault	16	5	
22272	0	Pronto	4	5	
22373	0	I/O	4	5	
22477	0	I/O	4	5	

Criar Suspendir Prosseguir Finalizar Prioridade PCB

Gerência de Me...

0	10	20	30	40	50	60	70	80	90
1	11	21	31	41	51	61	71	81	91
2	12	22	32	42	52	62	72	82	92
3	13	23	33	43	53	63	73	83	93
4	14	24	34	44	54	64	74	84	94
5	15	25	35	45	55	65	75	85	95
6	16	26	36	46	56	66	76	86	96
7	17	27	37	47	57	67	77	87	97
8	18	28	38	48	58	68	78	88	98
9	19	29	39	49	59	69	79	89	99

Tam. LPL 92

Tam. LPM 0

Gerência do Processador

Execução

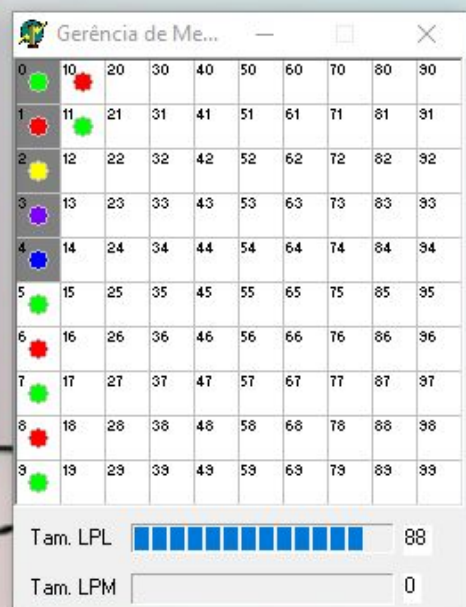
Pronto

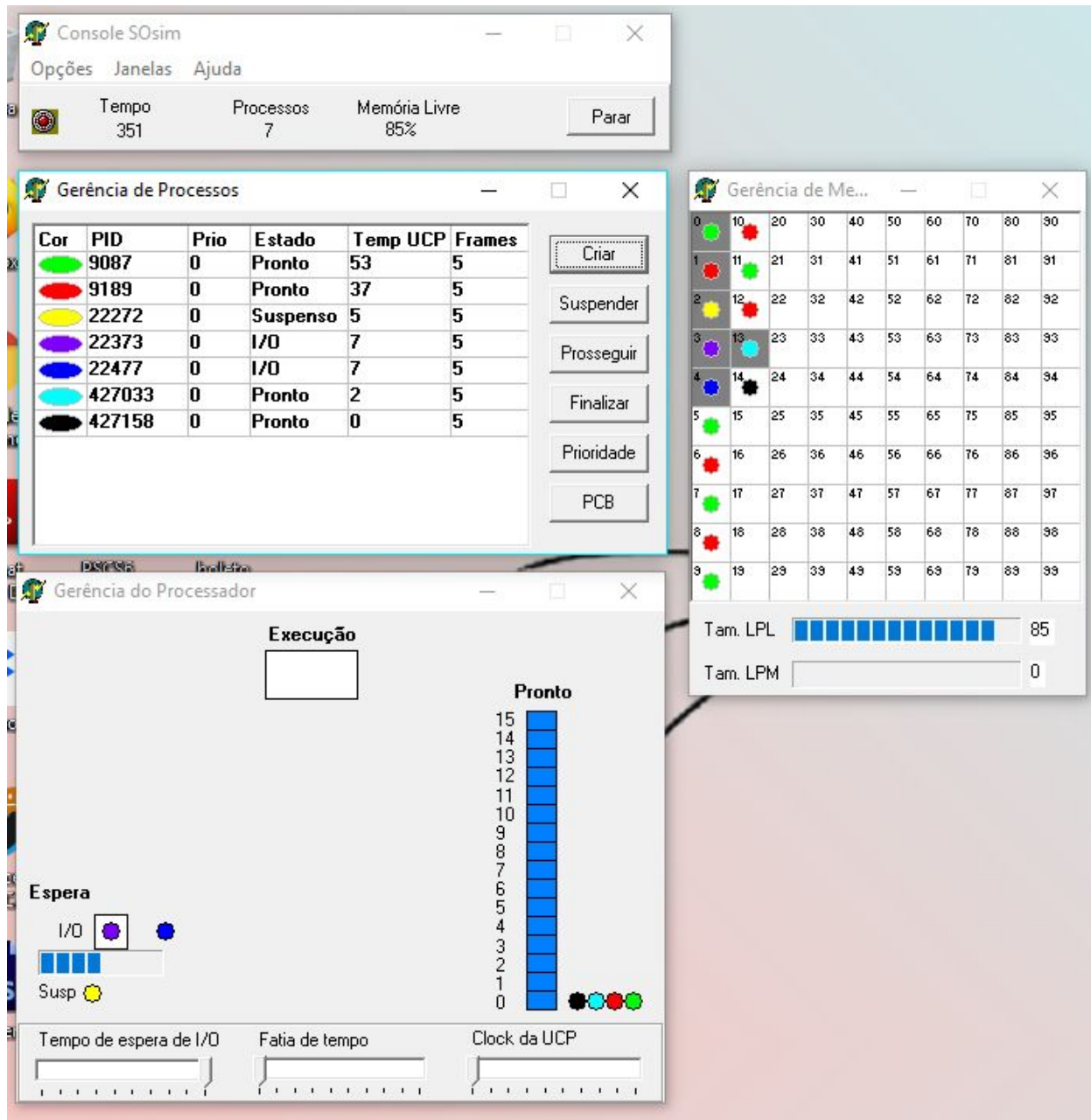
Espera

I/O Susp

Tempo de espera de I/O Fatia de tempo Clock da UCP







Questão teórica para responder com a ajuda do simulador:

- Quais os critérios utilizados pelo simulador para selecionar o processo a ser transferido para o arquivo de paginação (swap out)?

O processo é selecionado analisando qual processo possui menor chance de utilizar a CPU nos próximos instantes e assim ele manda esse processo para o arquivo de paginação.

- Quando o processo deve ser transferido novamente para a memória principal (swap in)?

Quando o limite de memória principal for referenciada, ou seja é utilizada a política de demanda, desta forma os processos só irão não para a memória principal quando forem realmente necessários.

[Questão-2] Com relação ao problema de Deadlock. Pesquisa e descreva o algoritmo do banqueiro (criado por Dijkstra) que pode ser utilizado para evitar impasses. Sempre que recursos são solicitados, o algoritmo avalia se atender à solicitação levará a um estado inseguro e se isso ocorrer, ela não é atendida. Adicionalmente, escreva o algoritmo do banqueiro em C/C++ e apresente alguns exemplos de sua execução.

O algoritmo funciona para solucionar um impasse explicado em que um banqueiro de uma pequena cidade pode negociar com um grupo de clientes para os quais ele libera linhas de crédito. E esse algoritmo verifica se a liberação de uma requisição é capaz de levar a um estado inseguro. É possível verificar se um estado é seguro ou não observando esta tabela abaixo. Estão disponíveis 7 créditos para serem usados pelos clientes e o banqueiro precisa gerenciar isso. Esse estado mostrado na tabela é seguro, pois, com essa quantidade de recursos disponíveis ele pode dar a um dos clientes a quantidade que ele precisa para atingir o crédito máximo e assim que ele terminar de usar os créditos eles são devolvidos e assim os outros podem usá-lo e desta forma todos conseguem obter o crédito. Caso o crédito disponível fosse, por exemplo, 3 não seria possível ele administrar isso para que todos pudessem usar, considerado assim um caso inseguro. E por fim caso a requisição do cliente seja dada como insegura ela será negada, caso contrário o recurso será liberado.