Memory exercises

Exercise 1

We have a virtual memory computer with paging. Virtual addresses are 32 bits and pages are 4 Kbytes. The physical memory of the computer is 16 Mbytes (each position is 1 byte).

a. How many page frames is physical memory divided into?
b. How many bits of the virtual address are used to identify the page?
c. How many bits are needed to address a page frame?
d. Suppose we have a single-level page table: How many entries does the page table have?

Solution 1

a)
If the page size is 4K, the frame size is too. If there is 16M of physical memory:

$$\frac{16 \times 1024 \times 1024}{4 \times 1024} = 4096 \text{marcos}$$

b)
If a page is 4K, 12 bits are required as an offset (or offset) in the virtual address (since 212 = 4096). Therefore, the bits of the virtual address that are used to identify the page are 32 - 12 = 20 bits. c) To address a frame, at least 12 bits are needed, considering the installed memory (4096 frames). (since 212 = 4096). d) With a one-level TP, the number of entries is the number of virtual memory pages: 220 = 1M entries.

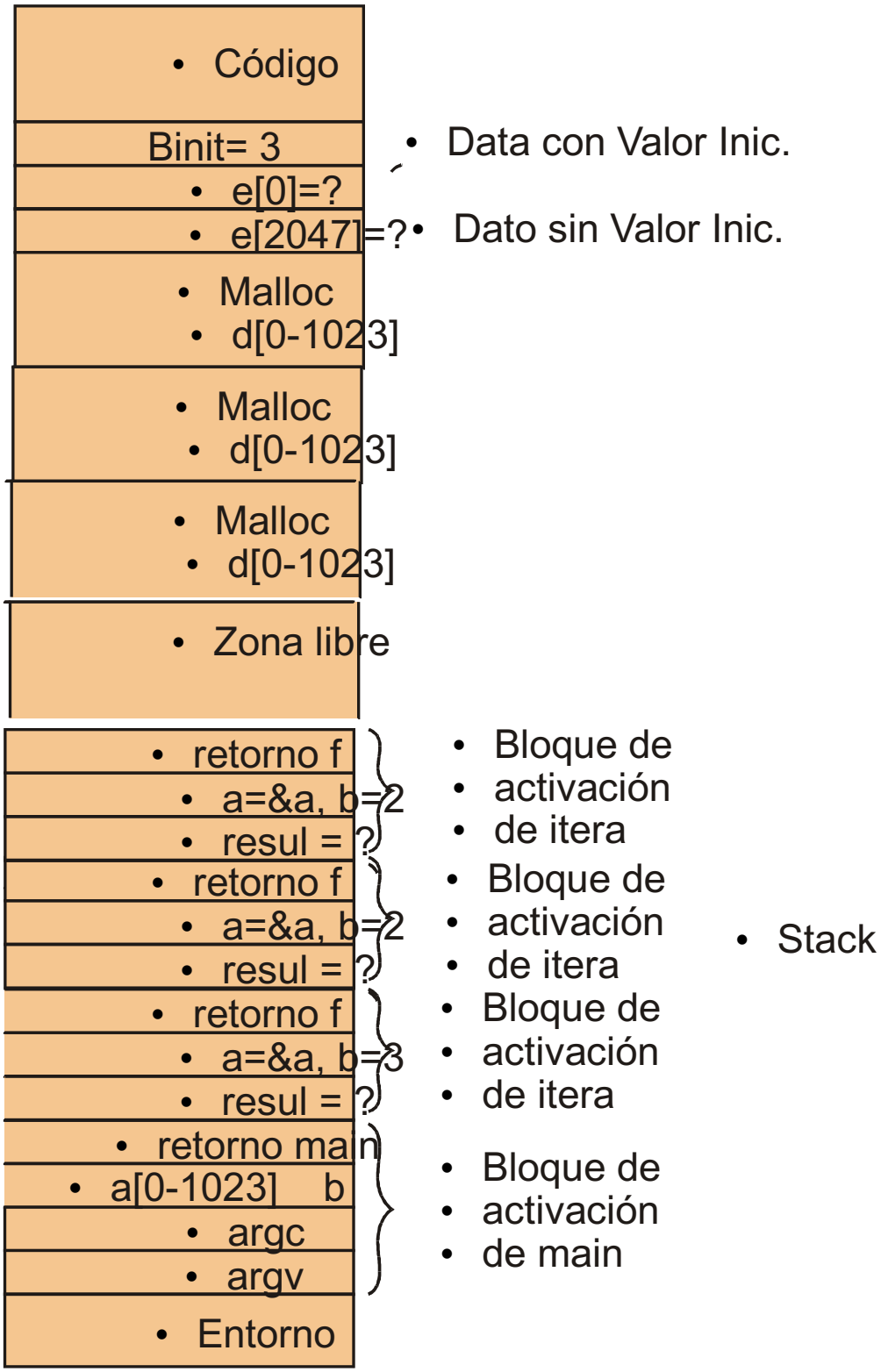
Exercise 2

Given the following code that executes three times the function iterates iteratively. Draw the memory map of the running program when the iterate function has been invoked the third time (it is verified that b = 0).

```
int Binit=3;
int e[2048];

int itera(int *a, int b)
{
  int *d;
  int resul;

  d=(int)malloc(1024,sizeof(int));

  b=b-1;
  if(b>0)
       resul=itera(a, b);
  else resul=0;

  resul=resul+1;

  return(resul);
}

void main(void)
{
  int a[1024];
  int b;

  b=Binit;
  resul=itera(a,b);
}
```

Solution 2

- Imagen de memoria



Exercise 3

In the previous operating system, the memory manager must allow us to be able to execute the 5 previous processes concurrently. For this, we have a disk space of 131,072 bytes (128K) and a physical memory of 32768 bytes (32K). We know that our processes when executed have the following parameters:

process	code	stack	data
A	4096	2000	2048
B	16384	8200	8192
C	2048	1000	1024
D	16384	8200	8192
E	16384	8200	8192

The data indicates the size in bytes of each of the segments that are part of the process image. Knowing that a page cannot contain parts of two different segments (stack, code or data), we have to determine the page size that our system should use and two options are considered: 4096-byte (4K) pages or 512-byte pages (1 / 2K).

Se pide:

1.- ¿ Which would be the most appropriate option, 4096 bytes or 512 bytes? Fully justify your answer by showing all the calculations you have needed to reach that conclusion.

2.- What is the format of each entry in the Page Table with the chosen page size? Justify the size of the fields with addresses. How many entries are there in the page table (rows)? You can add the bits that you consider necessary for the proper functioning of the system, indicating what they will be used for.

NOTE: if you have not been able to answer question 1, choose any of the two options, expressly indicate it and answer this question with that information.

3.- How many Page Tables will there be in this system?

Solution 3

1. **NP =** number of pages needed

process	code	NP4096	NP512	stack	NP4096	NP512	data	NP4096	NP512
A	4096	1	8	2000	1	4	2048	1	4
B	16384	4	32	8200	3	17	8192	2	16
C	2048	1	4	1000	1	2	1024	1	2
D	16384	4	32	8200	3	17	8192	2	16
E	16384	4	32	8200	3	17	8192	2	16

With pages of 4096 bytes we need to be able to execute the 5 processes load in memory:
1 + 4 + 1 + 4 + 4 + 1 + 3 + 1 + 3 + 3 + 1 + 2 + 1 + 2 + 2 = 33 pages
Since the system has an address space of 131072, the number of possible pages would be:
131072/4096 = $2^{17}/2^{12} = 2^5$ = 32 pages.
To be able to execute the processes we need 33 pages at least, and we only have capacity for 32.

With pages of 512 bytes we need to be able to execute the 5 load in memory:
8 + 32 + 4 + 32 + 32 + 4 + 17 + 2 + 17 + 17 + 4 + 16 + 2 + 16 + 16 = 219 pages
Since the system has an address space of 131072, the number of possible pages would be:
131072/512 = $2^{17}/2^9 = 28$ = 256 pages.
To be able to execute the processes we need 219 pages at least, and we have 256 then it is possible to execute these.

With pages of 512 bytes we need to be able to execute the 5 processes load in memory:
8 + 32 + 4 + 32 + 32 + 4 + 17 + 2 + 17 + 17 + 4 + 16 + 2 + 16 + 16 = 219 pages
Since the system has an address space of 131072, the number of possible pages would be:
131 072/512 = $2^{17}/2^9 = 2^8$ = 256 pages.
To be able to execute the processes we need 219 pages at least, and we have 256 then it is possible to execute these processes.

We will therefore choose pages of 512 bytes.

2.
The number of frames that we have in the system is:

32768 / 512 = 2^{15} / 2^9 = 2^6 we need 6 bits to reference each frame.

frame	P/A	R	M	RO	...
6 bits	1 bit	1 bit	1 bit	1 bit	1 bit

Each process has its own page table, then the number of entries in the page table will depend on which of the processes (A, B, C, D or E) we are referring to. For process A, we will need 16 entries, as many as pages are part of the process.

3. There would be at least 5 page tables, one for each process.

Exercise 4

Reading the following code:

```
float a;  float b=2;

void f1(int c) {
        float i;
        int j;

        b = i+ 5;
        .......
        f2(j)
        return;
}

void f2(int w) {
        static f = 2;
        a=a+w+f;
        .......
        return;
}

main (int argc, char **argv) {
        int *p;
        char *q;

        p = (int *) malloc (512)

        f1(b);

        q=mmap (fichero)
        .......
        free (p)
        .......
        free (q)
        ....
        exit (0)
}
```

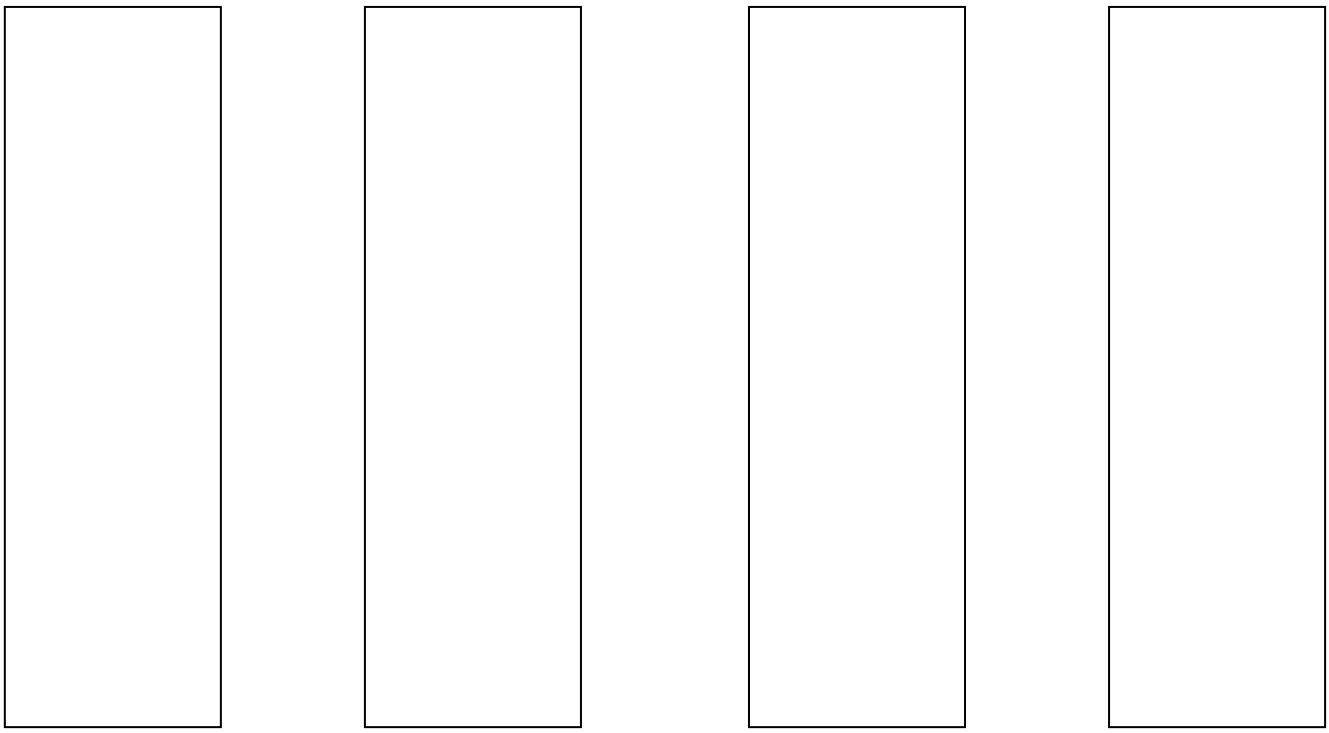
A

B

C

Explain how the memory regions or segments of the program evolve when its execution is launched, and in the points of the code indicated with arrows, filling in the following figures.

Draw the content of each segment.



Start of execution

A

B

C

Solution 4

• Start of execution:
- o Code,
- o Data with initial value (b = 2, f = 2),
- o Data without initial value (a =?),
- o Stack (var. Environment, argc, argv, return main, p = ?, q =?)

• A:
- o Code,
- o Data with initial value (b = 2, f = 2),
- o Data without initial value (a =?),
- o Stack (var. Environment, argc, argv, return main, q =?)

o Heap (p)

• B:
o Code,
o Data with initial value (b = 2, f = 2),
o Data without initial value (a =?),
o Stack (var. Environment, argc, argv, return main, q = ?,
o Activation block of f1 (i, j, c, return f1),
o Activation block of f2 (w, return f2)
o Heap (p)

• C:
o Code,
o Data with initial value (b = 2, f = 2),
o Data without initial value (a =?),
o Stack (var. Environment, argc, argv, return main),
o Heap (p)
o File projection (q)

Exercise 5

En un sistema de gestión de memoria, el sistema operativo ocupa 10 K y dispone de una memoria libre de 30 K, en la que se introducen los siguientes trabajos:

Process	Size	T. Stay in Memory
TI	4 K	0,3 seg.
T2	2 K	0,1 seg.
T3	7 K	0,5 seg.
T4	15 K	0,4 seg.
T5	8 K	0,8 seg.
T6	12 K	0,2 seg.

a) If the memory has 3 fixed partitions of 4K, 10K and 16K, and a FCFS scheduler is used with a single queue and allocation of the partition according to the criterion of the best available. Represent the states of said memory, indicating the average return time and the fragmentation in each state.
Repeat it with the allocation of the partition according to the criterion of only the one that best suits.
b) If memory management is done through the use of variable partitions, indicate how the Partition Description Table (PDT) would look, when entering all jobs.
Represent the various states through which the memory passes, calculating in each case the fragmentation and the average return time.

Solución

a)

Process	Size	T. Stay in Memory
TI	4 K	0,3 seg.
T2	2 K	0,1 seg.
T3	7 K	0,5 seg.
T4	15 K	0,4 seg.
T5	8 K	0,8 seg.
T6	12 K	0,2 seg.

	0	0,1	0,2	0,3	0,4	0,5	0,6
4 K	T1 FI=0	T1 FI=0	T1 FI=0	FE=4	FE=4	FE=4	
10 K	T2 FI=8	T5 FI=2	T5 FI=2	T5 FI=2	T5 FI=2	T5 FI=2	
16 K	T3 FI=9	T3 FI=9	T3 FI=9	T3 FI=9	T3 FI=9	T4 FI=1	

	0,6	0,7	0,8	0,9	1,0	1,1	1,2
4 K	FE=4	FE=4	FE=4	FE=4	FE=4	FE=4	
10 K	T5 FI=2	T5 FI=2	T5 FI=2	FE=10	FE=10	FE=10	
16 K	T4 FI=1	T4 FI=1	T4 FI=1	T6 FI=4	T6 FI=4	FE=16	

AVG TURNAROUND = 3,8/6 = 0,63

PROCESS	INPUT	OUTPUT	TOTALS
T1	0	0,3	0,3
T2	0	0,1	0,1
T3	0	0,5	0,5
T4	0	0,9	0,9
T5	0	0,9	0,9
T6	0	1,1	1,1

Memory exercises

TOTAL = 3,8

0 0,1 0,2 0,3 0,4 0,5 0,6

4 K	T1 FI=0	T1 FI=0	T1 FI=0	T2 FI=2	FE=4	FE=4
10 K	T3 FI=3	T3 FI=3	T3 FI=3	T3 FI=3	T3 FI=3	T5 FI=2
16 K	T4 FI=1	T4 FI=1	T4 FI=1	T4 FI=1	T6 FI=4	T6 FI=4

0,6 0,7 0,8 0,9 1,0 1,1 1,2

4 K	FE=4	FE=4	FE=4	FE=4	FE=4	FE=4
10 K	T5 FI=2	T5 FI=2	T5 FI=2	T5 FI=2	T5 FI=2	T5 FI=2
16 K	FE=16	FE=16	FE=16	FE=16	FE=16	FE=16

1,2 1,3 1,4 1,5 1,6 1,7 1,8

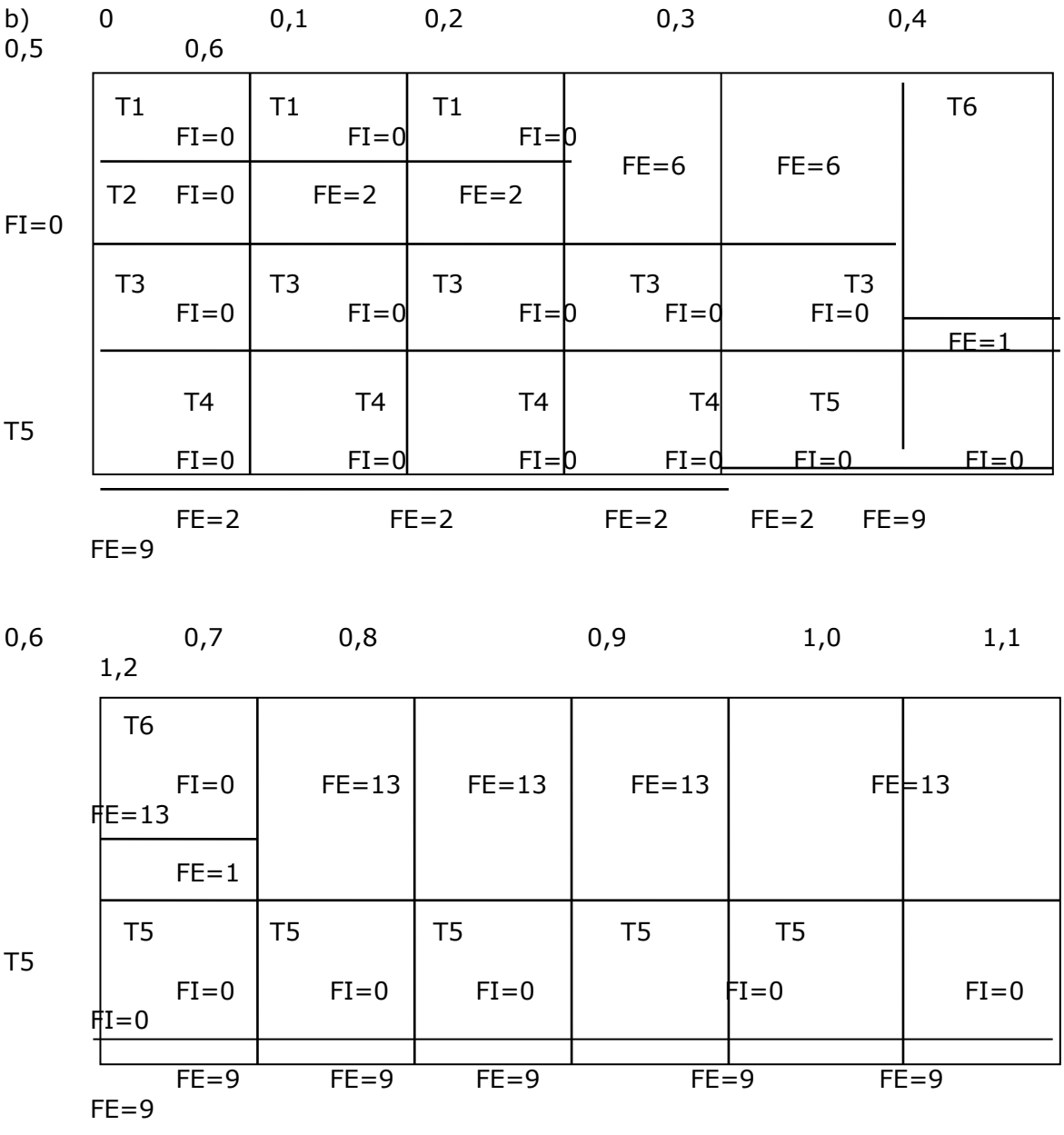
4 K	FE=4					
10 K	T5 FI=2					
16 K	FE=16					

AVG TURNAROUND = 3,5/6 = 0,58

PROCESS	INPUT	OUTPUT	TOTALS
T1	0	0,3	0,3
T2	0	0,4	0,4
T3	0	0,5	0,5
T4	0	0,4	0,4
T5	0	1,3	1,3
T6	0	0,6	0,6

TOTAL = 3,5

b)



PARTITION TABLE

```
N° PARTICIÓN   BASE   TAMAÑO   ESTADO
      0          0    12 K   ASIGNADA
      1         13     8 K   ASIGNADA
```

TURNAROUND

```
TRAB   ENT   SAL   TOT
T1     0     0,3   0,3
T2     0     0,1   0,1
T3     0     0,5   0,5
T4     0     0,4   0,4
T5     0     1,2   1,2
T6     0     0,7   0,7
           TOTAL   3,2
```

$$\text{AVG TURNAROUND} = \frac{3,2}{6} = 0,53$$

Exercise 6

A system manages its memory by the method of variable partitions. Assuming that the Partition Description Table (PDT), at a given time, has the following content:

PARTITONS	BASE	SIZE	STATUS
0	0	40 K	ALLOCATED
1	56	30 K	ALLOCATED
2	100	12 K	ALLOCATED
3	117	30 K	ALLOCATED

Assuming that the system has 170K of total memory, indicate where a 13K P1 program would be located and then another 5K P2 program using the three possible strategies separately.

Solución

TABLA DESCRIPCIÓN PARTICIONES TABLA FRAGMENTOS DISPONIBLES

N° PARTITION	BASE	SIZE	STATUS	ADDRESS	FREE SPACE	
0	0	40 K	ASIGNADA	40 K	16 K	⇒ A
1	56	30 K	ASIGNADA	86 K	14 K	⇒ B
2	100	12 K	ASIGNADA	112 K	5 K	⇒ C
3	117	30 K	ASIGNADA	147 K	23 K	⇒ D

FIRST FIT	BEST FIT	WORST FIT
P1 ⇒ A	P1 ⇒ B	P1 ⇒ D
P2 ⇒ B	P2 ⇒ C	P2 ⇒ A

Exercise 7

In memory management with variable partitions, the process of merging adjacent holes, when they are free, is called combination to form a larger hole.
Let us suppose a system with variable partitions that does not allow combination, that is, when a job is removed from memory, the partition where said job was located cannot be joined to other contiguous partitions, to form a larger partition.
With the previous assumption, if we have a memory that initially has 50 Kb free for programs, draw the states through which said memory passes to contain the following jobs, calculating the fragmentations in each case:

PROCESS	SIZE	TIME
T1	22 K	1,2 seg.
T2	9 K	2,3 seg.
T3	12 K	1,0 seg.
T4	10 K	1,3 seg.
T5	6 K	2,2 seg.
T6	4 K	3,0 seg.
T7	15 K	1,1 seg.
T8	3 K	2,0 seg.

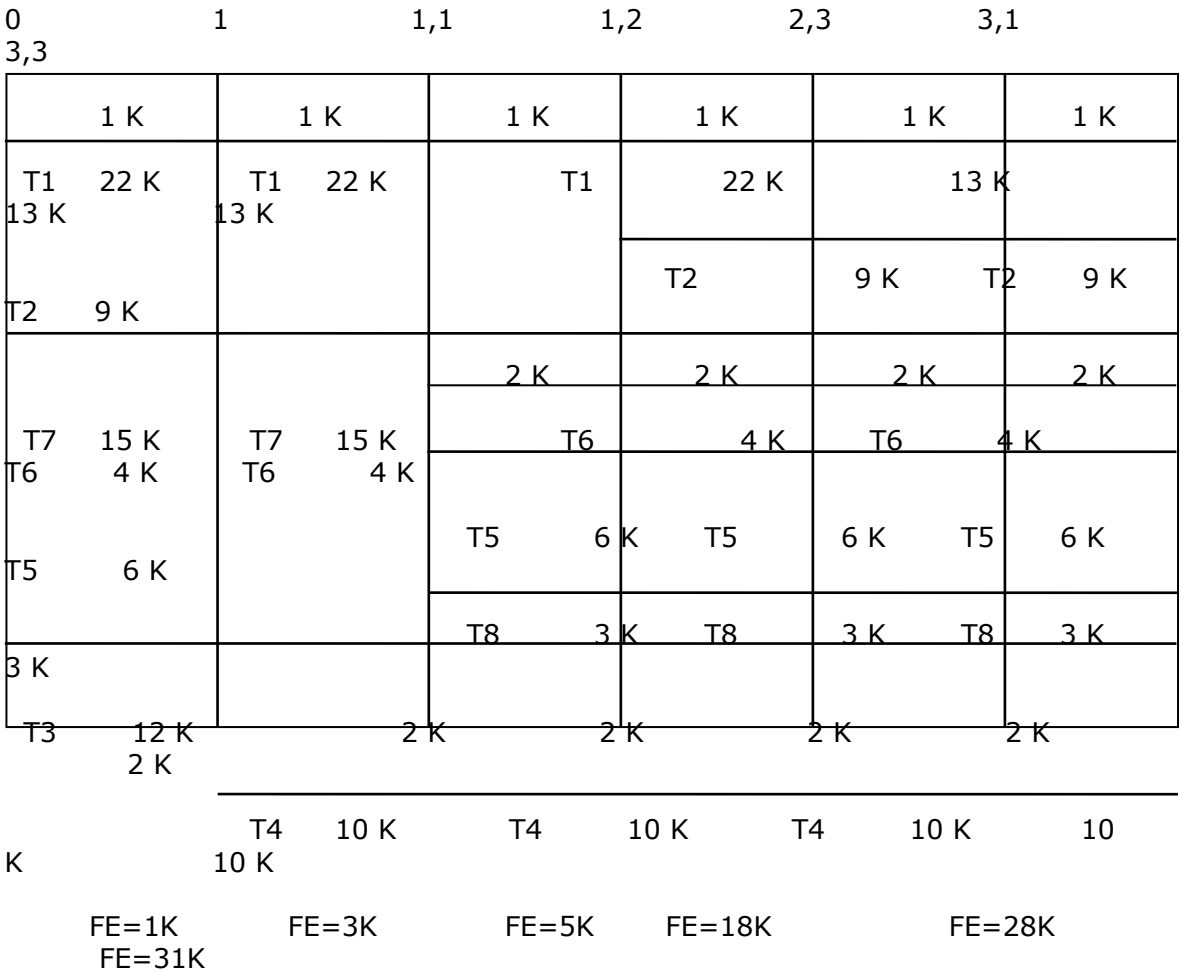
Jobs are managed using an SJF algorithm, which is an algorithm that assigns higher priority to jobs that spend less time in memory.
Jobs enter memory based on the best fit strategy.

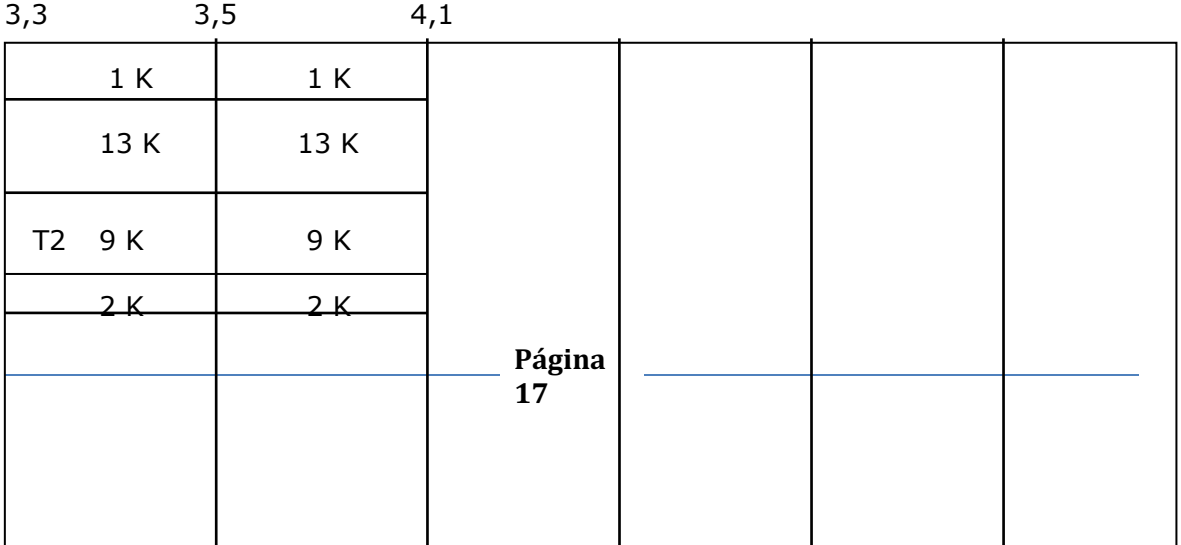
What is the average return or stay time?

Solución

Memory exercises

PROCESS	SIZE	TIME
T1	22 K	1,2 seg.
T2	9 K	2,3 seg.
T3	12 K	1,0 seg.
T4	10 K	1,3 seg.
T5	6 K	2,2 seg.
T6	4 K	3,0 seg.
T7	15 K	1,1 seg.
T8	3 K	2,0 seg.





T6 4 K	T6 4 K
6 K	6 K
3 K	3 K
2 K	2 K
10 K	10 K

FE=37 FE=46K

AVG TURNAROUND = 19,6/8 = 2,45

PROCESS	ARRIVAL	LEAVE	TOTALS
T1	0	1,2	1,2
T2	0	3,5	3,5
T3	0	1	1
T4	0	2,3	2,3
T5	0	3,3	3,3
T6	0	4,1	4,1
T7	0	1,1	1,1
T8	0	3,1	3,1

TOTAL = 19,6

Exercise 8

In a multiprogrammed system, there is a 100 Kb memory for programs and it allocates memory with a system of variable partitions, following the criteria of best fit. The job queue is managed by priorities, taking into account that the highest priority corresponds to the lowest number and is made up of the following jobs:

PROCESS	T. in MEMORY	PIORITY	SIZE	ARRIVAL T.
T1	0,2 msg.	1	10 K	0,3 msg.
T2	0,4 msg.	2	50 K	0,1 msg.
T3	0,1 msg.	1	70 K	0,2 msg.
T4	0,8 msg.	2	20 K	0,3 msg.
T5	0,7 msg.	3	80 k	0,1 msg.
T6	1,1 msg.	1	20 K	0,3 msg.

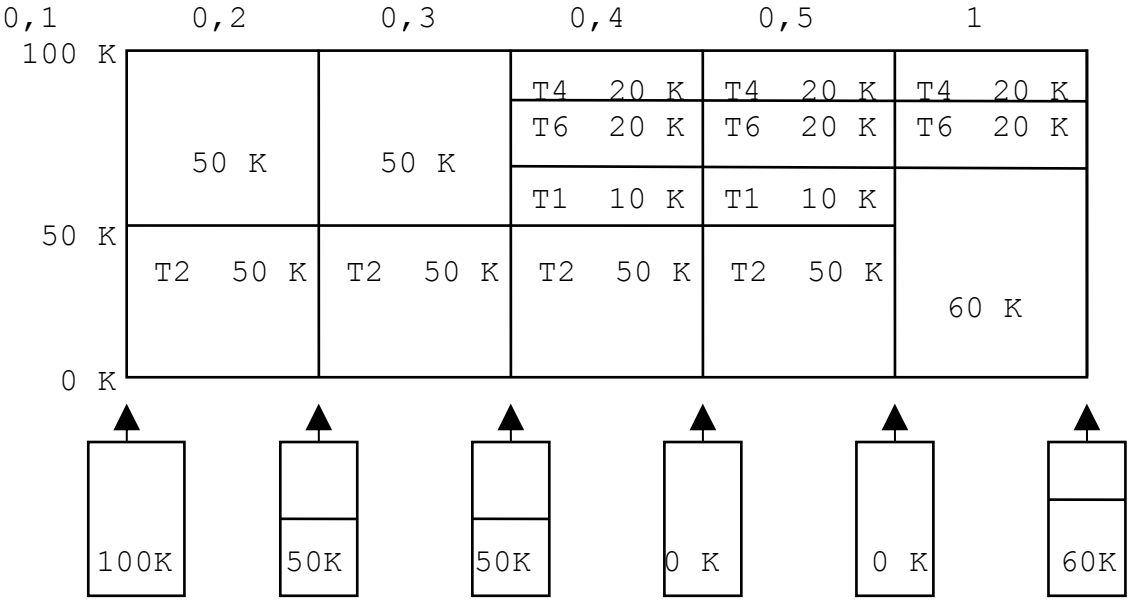
Taking into account these data:

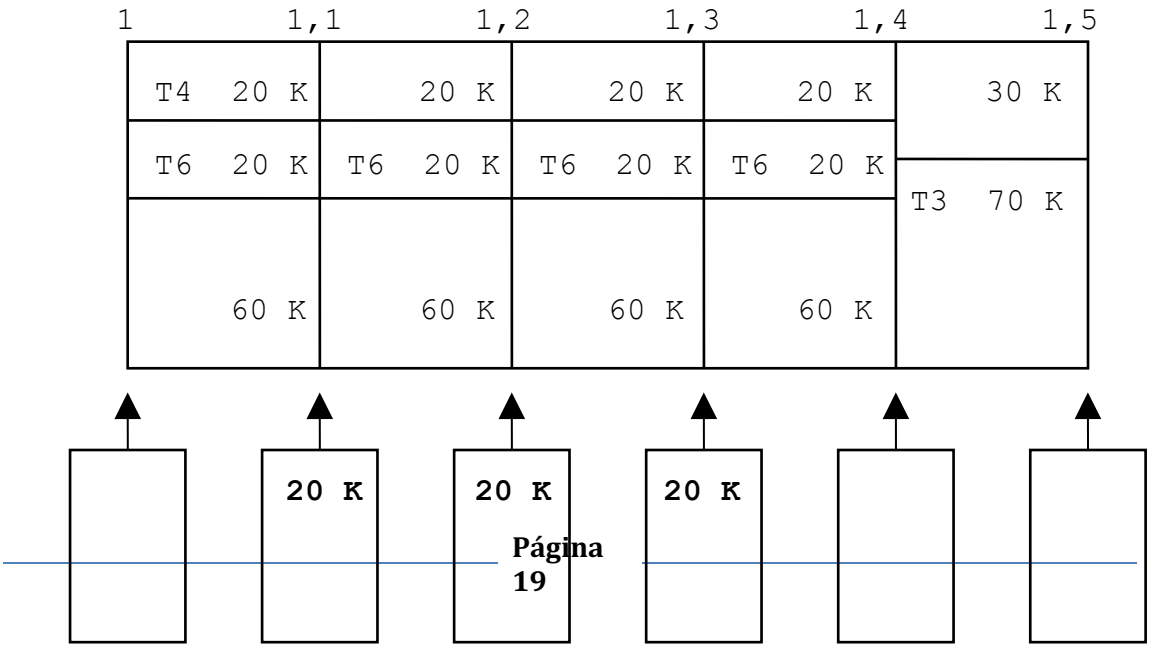
a) Specify the states through which the memory passes.
b) Represent the available fragment tables.

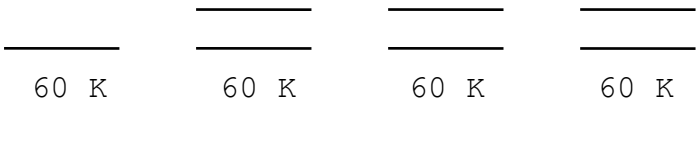
c) Indicate the status of the waiting queue before and after it undergoes modifications.
d) Calculate the average return time of the works.

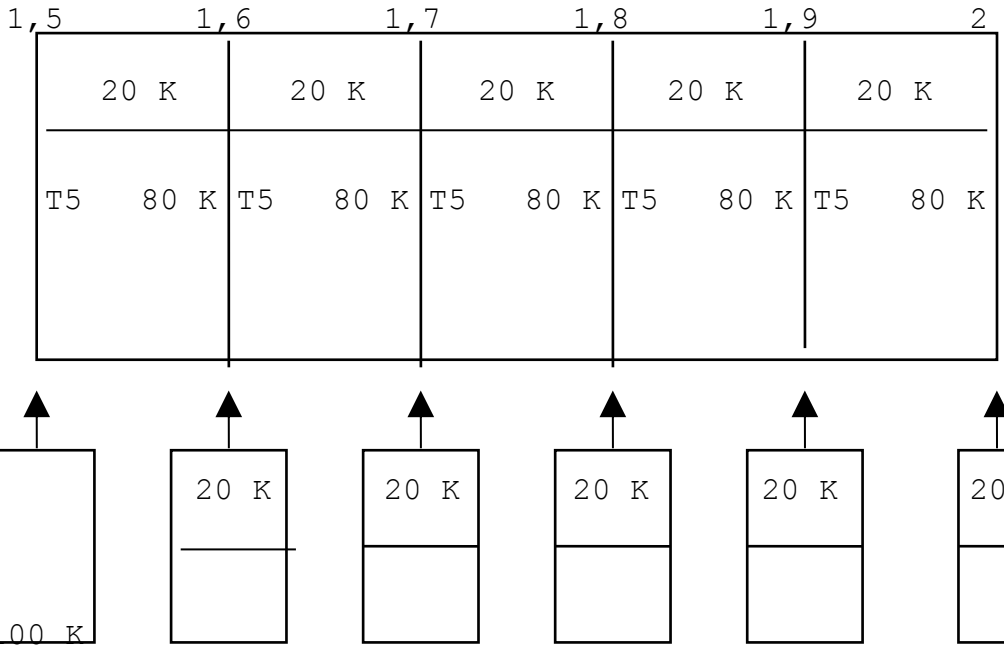
Solución

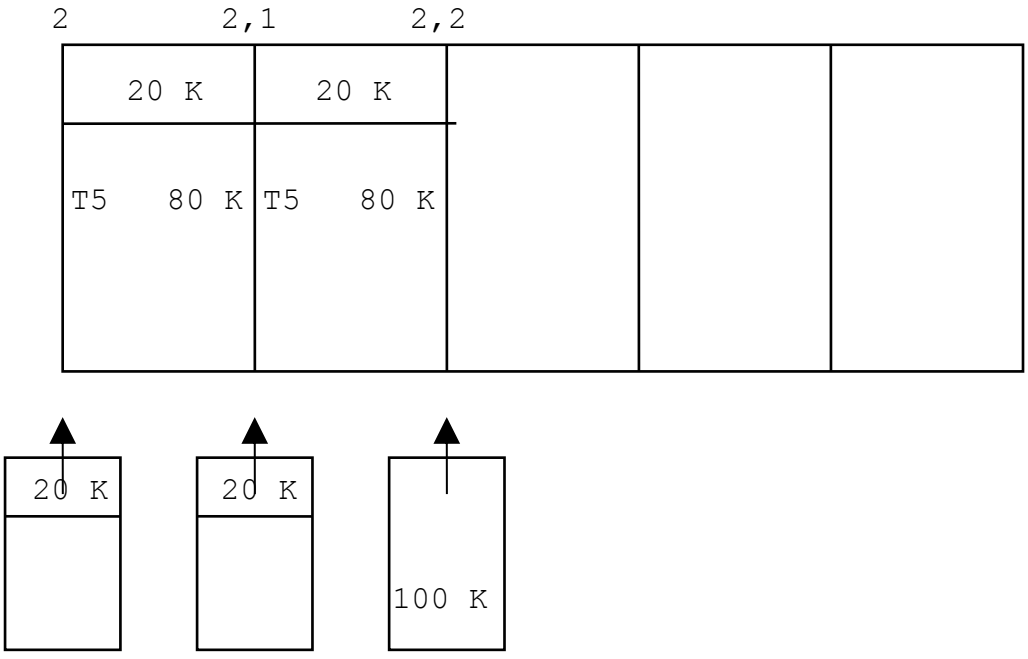
ARRIVAL	PROCESS	PRIORITY	SIZE	MEMORY T.
0,1msg.	T2	2	50 KB	0,4 msg.
0,1 msg.	T5	3	80 KB	0,1 msg.
0,2 msg.	T3	1	70 KB	0,1 msg.
0,3 msg	T1	1	10 KB	0,2 msg.
0,3 msg.	T6	1	20 KB	1,1 msg.
0,3 msg.	T4	2	20 KB	0,8 msg.











TIEMPO DE RETORNO T. MEDIO RET = 5,9/6 = 0,98

TRAB.	ENT.	SAL.	TOT	
T1	0,3	0,5	0,2	
T2	0,1	0,5	0,4	
T3	0,2	1,5	1,3	
T4	0,3	1,1	0,8	
T5	0,1	2,2	2,1	
T6	0,3	1,4	1,1	TOTAL = 5,9

ESTADO DE COLA DE ESPERA ANTES Y DESPUES DE MODIFICACIONES				
INSTANTE TIEMPO	C. DE ESPERA ANTES CARGA	C. DE ESPERA DESPUES CARGA	TRABAJOS MEMORIA	TRABAJO TERMINADO
0,1	T2,T5	T5	T2	
0,2	T5,T3	T3,T5	T2	
0,3	T3,T1,T6 T4,T5	T3,T5	T2,T1 T4,T6	
0,5	T3,T5	T3,T5	T4,T6	T1,T2
1,1	T3,T5	T3,T5	T6	T4
1,4	T3,T5	T5	T3	T6
1,5	T5		T5	T3
2,2				T5

Exercise 9

We have a 150 Kb memory to introduce programs, which is managed by variable partitions with automatic compaction and managing the entry of jobs into memory by priorities.
The strategy used is that of the best adjustment and compaction is carried out when the time it takes to do it is less than the time the next job has to wait to enter memory without compacting.
The system consumes 0.1 msg to move 1 Kb from one memory position to another and all jobs are assumed to be in the waiting queue at time 0 msg.

Job priorities are higher if they have a higher number associated with them.

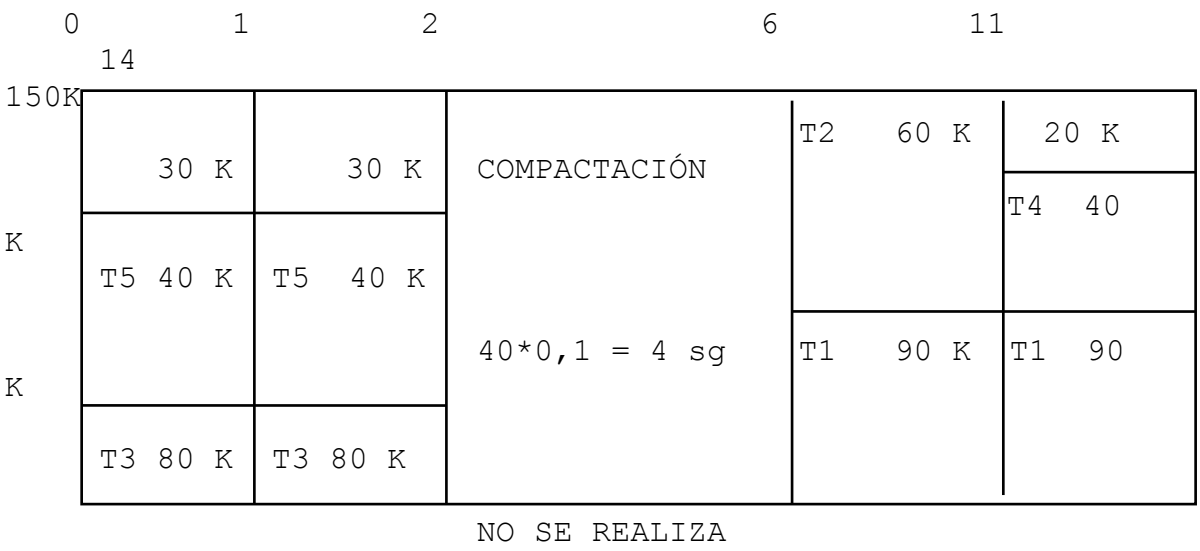
PROCESS	SIZE	MEMORY T.	PRIORITY
T1	90 K	8	3
T2	60 K	5	4
T3	80 K	2	1
T4	40 K	10	5
T5	40 K	6	2
T6	100 K	2	6

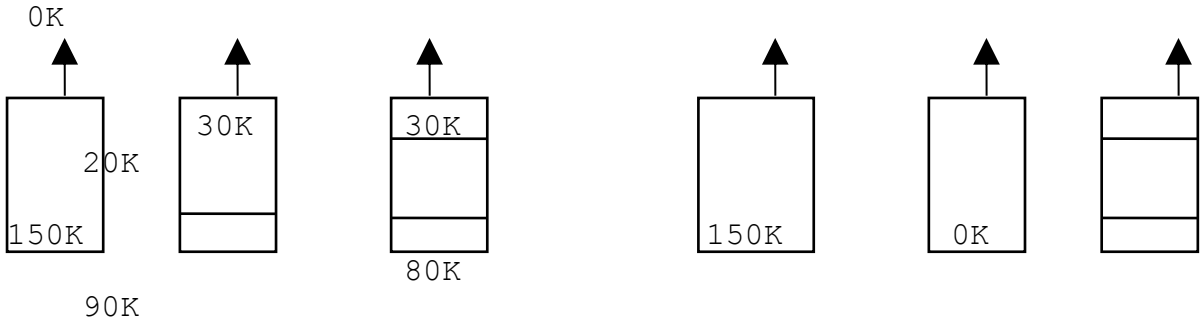
You want to solve the following points:

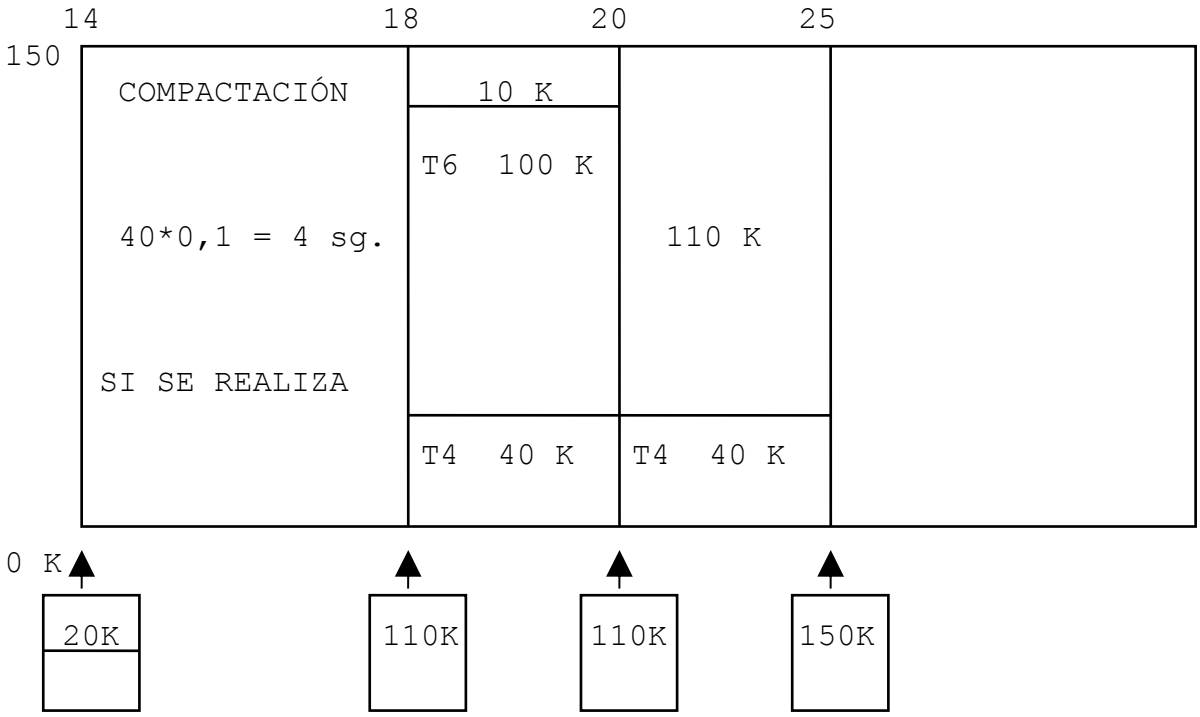
a) Represent the states through which the memory passes, specifying the moment in which the compaction is carried out.
b) Specify the areas of available fragments for each of the states.
c) Indicate the status of the waiting queue, before and after it undergoes modifications.
d) Calculate the average return time of the works.

Solución

PROCESS	SIZE	MEMORY T.	PRIORITY
T1	90 KB	8 seg.	3
T2	60 KB	5 seg.	4
T3	80 KB	2 seg.	1
T4	40 KB	10 seg.	5
T5	40 KB	6 seg.	2
T6	100 KB	2 seg.	6


0
1
2
6
11
14
150K
30 K
30 K
COMPACTACIÓN
T2 60 K
20 K
T4 40
K
T5 40 K
T5 40 K
40*0,1 = 4 sg
T1 90 K
T1 90
K
T3 80 K
T3 80 K
NO SE REALIZA


0K
20K
150K
30K
30K
80K
150K
0K
90K


14
18
20
25
150
COMPACTACIÓN
10 K
T6 100 K
40*0,1 = 4 sg.
110 K
SI SE REALIZA
T4 40 K
T4 40 K
0 K
20K
110K
110K
150K

90

AVERAGE TURNAROUND

PROCESS	ARRIVAL	LEAVE	TURNAROUND
T1	6 seg.	14 seg.	8 seg.
T2	6 seg.	11 seg.	5 seg.
T3	0 seg.	2 seg.	2 seg.
T4	11 seg.	25 seg.	14 seg.
T5	0 seg.	6 seg.	6 seg.
T6	18 seg.	20 seg.	2 seg.

TOTAL 37 sg.

$$\text{AVG TURNAROUND} = \frac{37}{6} = 6{,}16 \text{ sg.}$$

ESTADO DE COLA DE ESPERA ANTES Y DESPUES DE MODIFICACIONES

INSTANTE TIEMPO	C. DE ESPERA ANTES CARGA	C. DE ESPERA DESPUES CARGA	TRABAJOS MEMORIA	TRABAJO TERMINADO
0	T3,T5,T1, T2,T4,T6	T1,T2,T4,T6	T3,T5	
2	COMPACTACIÓN	T1,T2,T4,T6	T5	T3
6	T1,T2,T4,T6	T4,T6	T1,T2	T5
11	T4,T6	T6	T1,T4	T2
14	T6	COMPACTACIÓN	T4	T1
18	T6		T4,T6	
20			T4	T6
25				**T4**