SISTEMAS OPERATIVOS
ENTREGABLE S-13

MEMORIA

TRADUCTOR DE DIRECCIONES LÓGICAS A FÍSICAS

Daniel García Navarro.

José Ángel Pardo Cerdán.

Pablo Martínez Mondéjar.

ÍNDICE

ÍNDICE DE ILUSTRACIONES

Introducción

Cuando se desea ejecutar un programa, no se conoce la dirección de memoria en la que éste se ejecutará por lo que se necesita la ayuda de un elemento adicional, el cual es el sistema operativo como Gestor de Memoria.

Cuando los programas están en proceso de ejecución, crean direcciones lógicas de memoria creando la necesidad de tener un traductor de direcciones lógica a direcciones físicas, que son las direcciones que realmente referencian a memoria principal. [1]

El código creado se encarga de realizar la traducción de las direcciones lógicas a direcciones física para los métodos de gestión de memoria paginación, particionamiento y segmentación. [2]

1. Descripción de las estructuras de datos utilizadas

En el programa se han utilizado tres estructuras de datos para almacenar los datos según el tipo de método de gestión vistos: particionamiento, paginación y segmentación. Estas tres estructuras de datos forman parte de los datos de la estructura proceso, en la cual se agrupan todos los datos pertenecientes a un proceso: [3]

1.1 Particionamiento: estas variables almacenan los datos generales.

- dir_logica: almacena el valor de la dirección lógica.
- dir_fisica: almacena el valor de la dirección física.
- regBase: almacena el valor del registro base.
- regLimite: almacena el valor del registro límite.

Formato de la estructura en C:

```
Struct particionamiento {
        Int dir_logica;
        Int dir_fisica;
        Int regBase;
        Int regLimite;
};
```

1.2 Segmentación: estas variables almacenan los datos de la tabla de segmentación.

- nFilas: almacena el valor del número de filas que tiene la tabla de segmentos.
- base[maxSeg]: al macena el valor de la tabla de segmentos en la posición maxSeg.
- longitud[maxSeg]: almacena el valor de la longitud en la posición maxSeg.
- segmento[maxSeg]: almacena el valor del segmento en la posición maxSeg.
- RBTS: almacena el valor del Registro Base de la tabla de segmentos.
- RLTS: almacena el valor del Registro Limite de la tabla de segmentos.

Formato de la estructura en C:

```
Struct segmentacion {
        int nFilas;
        int base[maxSeg];
        int longitud[maxSeg];
        int segmento[maxSeg];
        int RBTS;
        int RLTS;

};
```

1.3 Paginación: estas variables almacenan los datos de la tabla de página.

- nFilas: almacena el valor del número de filas que tiene la tabla de páginas.
- pagina[maxPag]: almacena el valor del número de página de la tabla de página.
- marco[maxPag]: almacena el valor del marco en la posición maxSeg.
- RBTP: almacena el valor del registro base de la tabla de página.
- RLTP: almacena el valor del registro límite de la tabla de página.
- Tam_pagina: almacena el valor del tamaño de página.

Formato de la estructura en C:

```
Struct paginacion {
        int nFilas;
        int pagina[maxSeg];
        int marco[maxSeg];
        int tam_pagina;
        int RBTP;
        int RLTP;

};
```

1.4 Proceso: en esta estructura se engloban todas las estructuras anteriores, asignándolas a cada proceso:

- particionamiento p: se crea el proceso p a partir de la estructura particionamiento.
- segmentacion seg: se crea el segmento seg a partir de la estructura segmentación.
- paginacion pag: se crea una nueva página pag a partir de la estructura paginación.

Formato de la estructura en C:

```
Struct proceso {
        struct particionamiento p;
        struct segmentacion seg;
        struct paginacion pag;
};
```

2. Descripción general del código.

El código se basa en un main() principal donde se van a ejecutar siete funciones que se explicarán posteriormente: "cargarProcesos" , "mostrar", “LeerFichDirecciones”, “NumDireccionesLogicas”, “resolver”, “formato”, “dirtoa”.

2.1 Función principal: **int main(int argc, char** argv)**

Esta función es la encargada de mostrarnos un menú en el cual podemos seleccionar la opción deseada. Según la opción que se elija, se llama a una función o a otra.

Si queremos cargar los datos almacenados en el fichero se llama a la función **cargarProcesos()** la cual realizará su función.

Si queremos mostrar los datos almacenados, se llamará a la función **mostrar()** la cual realizará su función.

2.2 Función: **void cargarProcesos(struct proceso proces[],int max,char ruta[])**

A esta función se le pasa como parámetro un array de procesos y el valor del número máximo de procesos que vamos a tener que leer.

Descripción:

Al comienzo de esta función, se inicializan las variables auxiliares que se utilizarán posteriormente.

Seguidamente se abre el fichero donde se encuentran los datos a leer y se abre un bucle que no termina hasta que el puntero del fichero abierto apunta al final del mismo.

Dentro de este bucle encontramos un switch, en el que según la primera letra de cada línea leída del fichero de texto, se selecciona en que variable almacenar los datos que le sigue a esa letra, en su estructura de datos correspondiente.

Los casos que se pueden dar en este switch son:

- **caso 't':** si el programa encuentra una línea cuya primera letra es una t, el número que le sigue es almacenado en la variable tam_pag (tamaño de página) dentro de la estructura paginación.

- **caso 'i':** si el programa encuentra una línea cuya primera letra es una i, quiere decir que en las línea siguiente se encuentran los siguientes datos ordenados en este mismo orden: registro base, registro límite, RBTS, RLTS, RBTP, RLTP.
  Para hacer la lectura de estos datos utilizamos la variable "si_tabla" igualando a tres para que en la siguiente vuelta del bucle, entre en el caso 3 del switch que está en el caso default del primer switch, donde se almacenan todos los datos leídos anteriormente en sus variables correspondientes.

- **caso 'p':** Se sigue el mismo procedimiento que en el caso i solo que esta vez igualamos la variable "si_tabla" a 1 lo que hace que en el switch alojado en el caso default entre en el caso 1 en el que se almacenan los datos de la tabla paginación.
  Posteriormente incrementamos la variable " nProcPag" que es utilizada para indicar a qué proceso del array de procesos se le almacenan los datos de la tabla de paginación.

- **caso 's':** En este caso los siguientes datos serán los referentes a la tabla de segmentación. Aquí como en los dos casos anteriores igualamos la variable "si_tabla" a 2 para que en el switch del default entre en el caso 2, que es el caso donde se almacenan estos datos en sus variables correspondientes de la estructura segmentación.

- **caso '#':** Cuando la siguiente línea leída es un asterisco quiere decir que se ha terminado de definir una tabla, por lo que se modifica la variable "si_tabla" igualándola a 0.

Una vez leído todo el documento se cierra el fichero.

2.3 Función: **LeerFichDirecciones(char ruta[],char **DireccionesLog)**

Descripción:

Esta función se encarga de leer el fichero donde se encuentran las direcciones lógicas a convertir. Dichas direcciones serán almacenadas en la matriz de caracteres DireccionesLog.

Dentro de la función se abre el fichero en modo lectura y de tipo texto, y se lee línea a línea mientras el fichero no sea vacío.

Cuando se acaba de leer todas las líneas, se cierra el fichero.

2.4 Función: **int NumDireccionesLogicas(char ruta [])**

Descripción:

Esta función se encarga de contar el número de direcciones lógicas contenidas en el fichero.

Dentro de la función se abre el dichero en modo lectura y se comprueba si dicho fichero existe. Si el fichero existe se cuenta el número de dirección es lógicas y si no existe se muestra por pantalla el correspondiente error.

Finalmente se cierra el fichero.

2.5 Función: **void resolver(int Ndirecciones,char **direccionesLog, struct proceso p[],int ***resultados)**

Esta función se encarga de convertir las direcciones lógicas a direcciones físicas.

Variables:

int d=0,i=0,j=0: son variables índice donde “d” es el índice de DireccionLogica, “i” es el índice de proceso y “j” es un índice auxiliar.

Int marco: es la variable donde se almacena el número de marco.

Int dirección: es la variable donde se almacena el número de dirección de la tabla de segmentación.

Int base, longitud: son las variables donde se almacenan el valor base y longitud de la tabla de segmentación.

int dirlogica[2]: es un array auxiliar de tipo entero utilizado para operar con la direcciones cuyo formato es (pagina/segmento,desplazamiento).

Descripción:

Dentro de la función hay un switch que dependiendo del formato de la dirección lógica a convertir se realiza un caso u otro.

El primer caso es para cuando la dirección lógica a convertir es simple.

El segundo caso es para cuando la dirección lógica tiene formato (pagina/segmento, desplazamiento)

En el último caso se da la opción de que el formato sea erróneo. En este caso almacenamos un -3 en los resultados para dar a entender que no se ha podido resolver la dirección dada.

2.6 Función: **int formato(char dir[])**

Descripción:

Esta función comprueba el formato de las direcciones.

Si el formato de la dirección es simple, la función devuelve un 1, y un 2 si es del tipo (pagina/segmento, desplazamiento).

Además comprueba que el formato no sea erróneo, en cuyo caso devolvería un 0.

2.7 Función: **void dirtoa(char dir[], int *dirlogica)**

Descripción:

Esta función se encarga de construir un array de dos posiciones a partir de una dirección del tipo (pagina/segmento, desplazamiento).

2.8 Función: **void guardar(int *** resultados,char **direccionesLog,int Ndirecciones)**

Descripción:

Esta función crea el nombre de un fichero, seguidamente abre el fichero en modo lectura y de tipo texto, y se lee la primera línea del texto.

Finalmente se escriben los resultados y se cierra el fichero.

3. Descripción del formato de los ficheros de entrada y de salida.

3.1 Ficheros de entrada.

Los datos se han estructurado mediante la utilización de 3 tablas diferentes, una para almacenar los datos de las tablas de páginas, otra para los datos de la tabla de segmentos y otra para almacenar los datos de información de un proceso.

Para que la lectura de los datos sea correcta la estructura de estas tablas deben quedar reflejados de la siguiente manera:

[letra identificadora de tabla]

datos

-----→ (carácter identificador de fin de tabla)

Siendo la letra identificadora de tabla una de las siguientes posibles:

- **t:** hace referencia a un dato que almacena el valor del tamaño de la página.
  En el fichero de texto debe quedar escrita de la siguiente manera:

  t tam_pagina

- **i:** hace referencia a una tabla donde se almacenan los siguientes datos: Registro base, Registro límite, RBTS, RLTS, RBTP, RLTP.
  En el fichero de texto debe quedar escrita de la siguiente manera:

  i
  R.Base R.Limite RBTS RLTS RBTP RLTP ← Proceso 0
  R.Base R.Limite RBTS RLTS RBTP RLTP ← Proceso 1
  #

- **p:** hace referencia a una tabla donde se almacenan los siguientes datos: página y marco.
  En el fichero de texto debe quedar escrita de la siguiente manera:

  p ← Proceso 0
  pagina[0] marco[0]
  pagina[1] marco[1]
  pagina[2] marco[2]
  .
  .
  .
  pagina[n] marco[n]
  #

  p ← Proceso 1
  pagina[0] marco[0]
  pagina[1] marco[1]
  pagina[2] marco[2]
  .
  .
  .
  pagina[n] marco[n]
  #

  Pueden existir diferentes tablas de páginas para diferentes procesos.

- **s:** hace referencia a un tabla donde se almacenan los siguientes datos: segmento, base y longitud.
  En el fichero de texto debe quedar escrita de la siguiente manera:

  s
  segmento[0] base[0] longitud[0]
  segmento[1] base[1] longitud[1]
  segmento[2] base[2] longitud[2]
  .
  .
  .
  segmento[n] base[n] longitud[n]
  #

3.2 Ficheros de salida

Los datos resultantes de la traducción de las direcciones lógicas a físicas se guardan en un fichero, existiendo 3 ficheros (uno por ejercicio), estructurados de la siguiente manera:

DIRECCION: (valor dirección lógica)

- Particionamiento: (valor dirección física)
- Paginación: (valor dirección física)
- Segmentación: (valor dirección física)

.

.

.

.

DIRECCION: (valor dirección lógica)

- Particionamiento: (valor dirección física)
- Paginación: (valor dirección física)
- Segmentación: (valor dirección física)

Existen tantos resultados como direcciones lógicas.

Además por cada proceso se crea un fichero con sus resultados con la estructura descrita.

Estas soluciones de direcciones físicas pueden ser o un número, un ERROR o una INCOMPATIBILIDAD.

Un ejemplo de esta estructura la muestra la Ilustración 1.

```
PROCESO: P0

DIRECCION: 1
    - Particionamiento: 28
    - Paginacion: 808
    - Segmentacion: Incompatible
-------------------------------------------------
DIRECCION: 123
    - Particionamiento: Error
    - Paginacion: 984
    - Segmentacion: Incompatible
-------------------------------------------------
DIRECCION: (1,23)
    - Particionamiento: Incompatible
    - Paginacion: 123
    - Segmentacion: 501
-------------------------------------------------
DIRECCION: (4323,70)
    - Particionamiento: Incompatible
    - Paginacion: Error
    - Segmentacion: Error
-------------------------------------------------
DIRECCION: 12
    - Particionamiento: 39
    - Paginacion: 896
    - Segmentacion: Incompatible
-------------------------------------------------
```

Ilustración 1: Estructura de fichero de salida

4. Resultados de la ejecución del programa con los ficheros de entrada de los ejercicios 2, 3 y 4.

4.1 Muestreado de los datos del ejercicio 1

En la ilustración 2 se muestra por pantalla los resultados de la lectura del fichero del ejercicio 1.

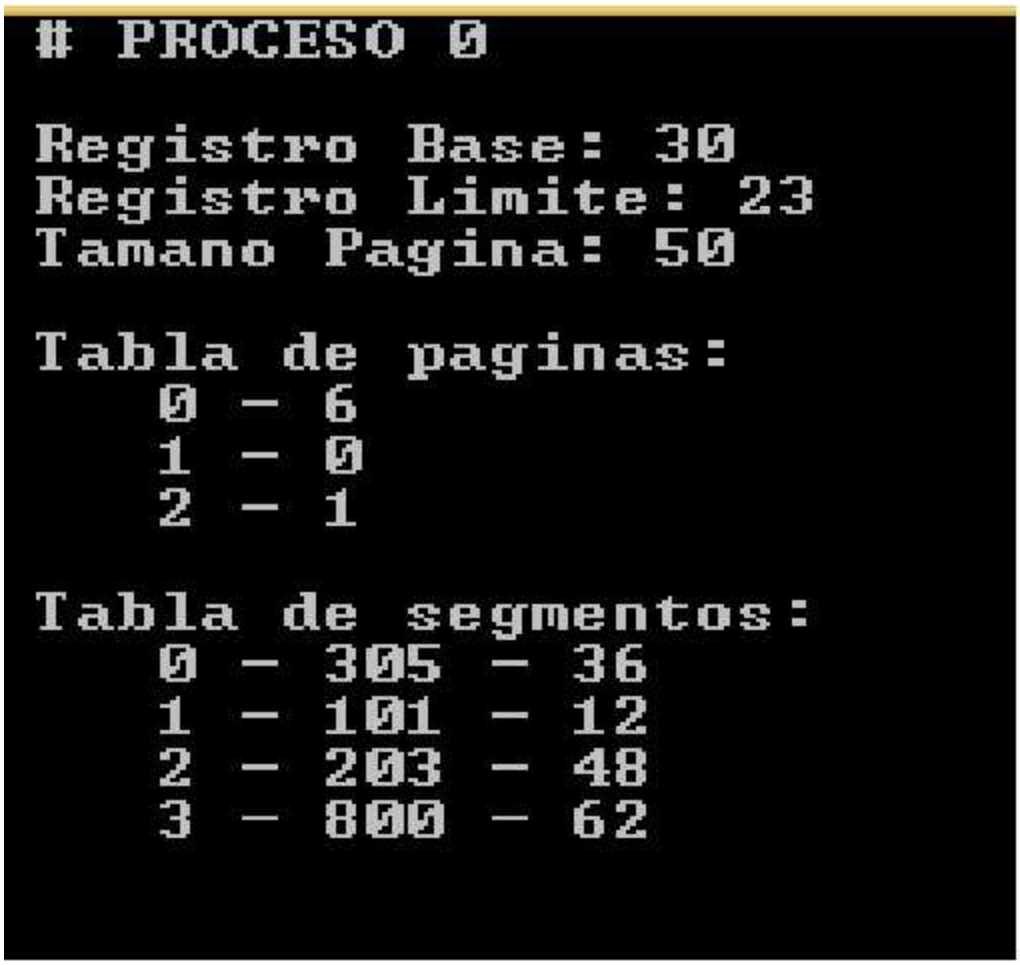

Ilustración 2: Muestreado de datos del ejercicio 1

Estructura del fichero leído:

t 50

i

30 23 -1 -1 -1 -1

#

p

0 6

1 0

2 1

#

s

0 305 36

1 101 12

2 203 48

3 800 62

#

4.2 Muestreado del ejercicio 2:

En la ilustración 3 se muestra por pantalla los resultados de la lectura del fichero del ejercicio 2

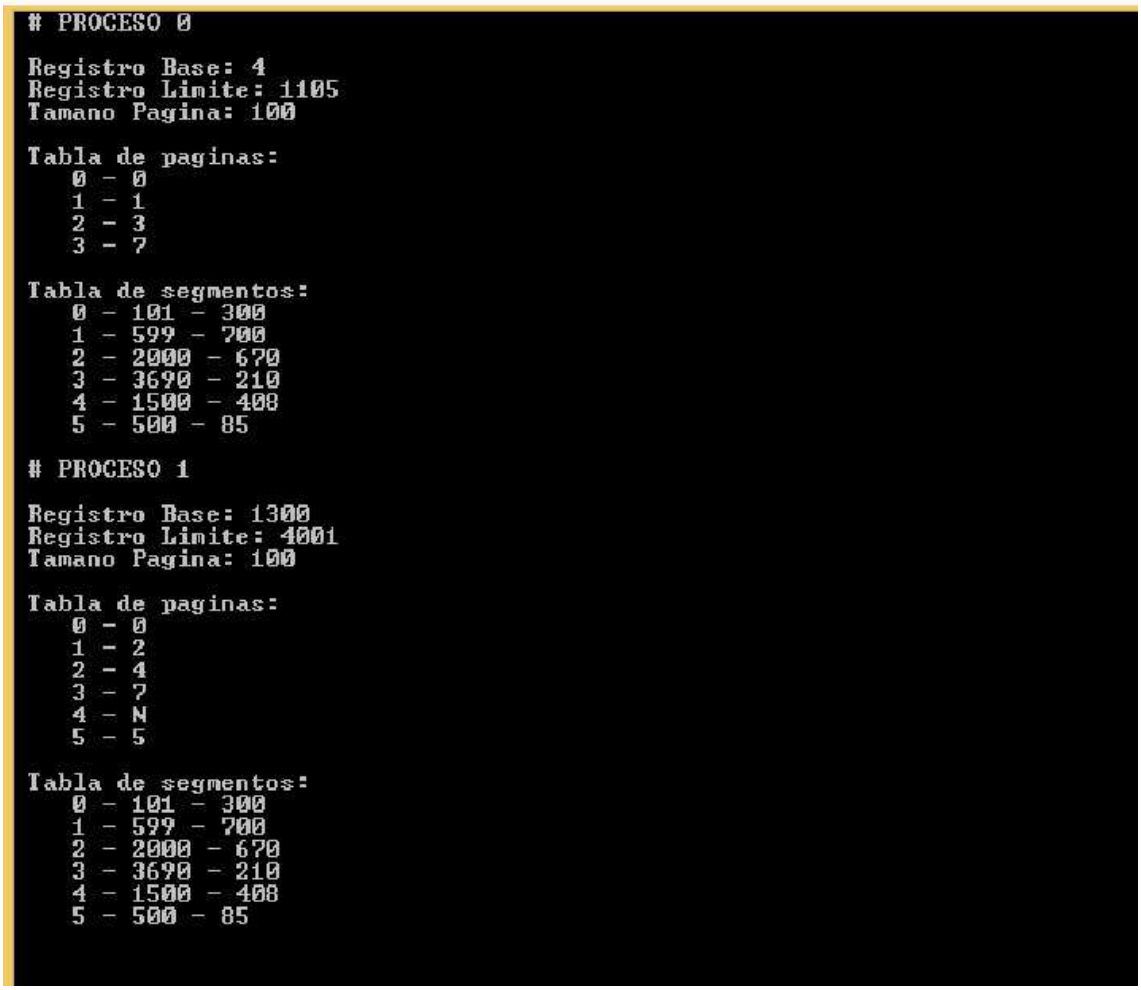

Ilustración 3: Muestreado de datos del ejercicio 2

Estructura del fichero leído:

t 100

i

4 1105 3 2 0 0

1300 4001 0 3 0 0

#

p

0 0

1 1

2 3

3 7

#

p

0 0

1 2

2 4

3 7

4 -1

5 5

#

s

0 101 300

1 599 700

2 2000 670

3 3690 210

4 1500 408

5 500 85

#

4.3 Muestreado del ejercicio 3:

En la ilustración 4 se muestra por pantalla los resultados de la lectura del fichero del ejercicio 3.

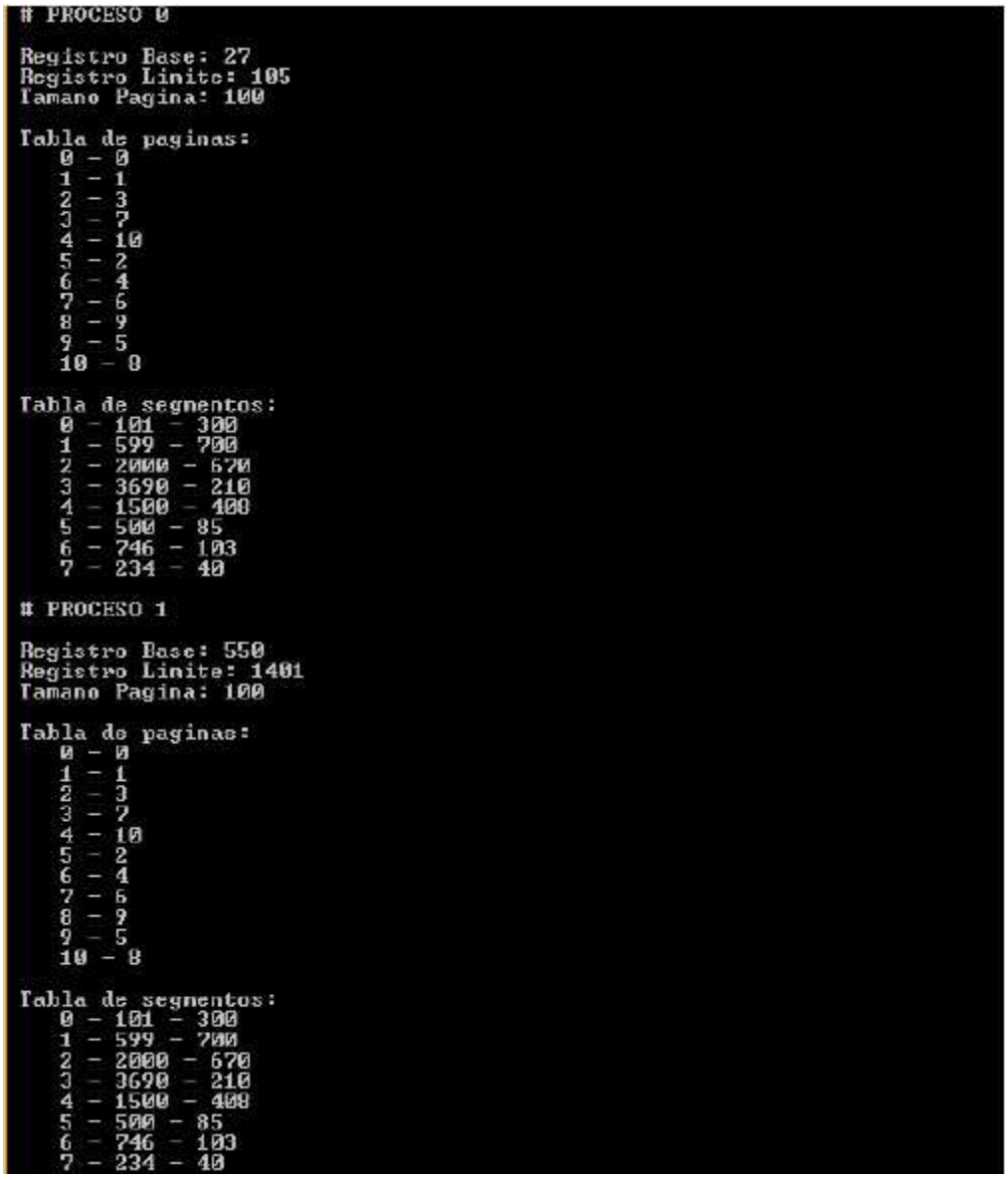

Ilustración 4: Muestreado de datos del ejercicio 3

Estructura del fichero leído:

t 100

i

27 105 4 3 0 3

550 1401 0 3 4 6

#

p

0 0

1 1

2 3

3 7

4 10

5 2

6 4

7 6

8 9

9 5

10 8

#

s

0 101 300

1 599 700

2 2000 670

3 3690 210

4 1500 408

5 500 85

6 746 103

7 234 40

#

5. Conclusión

Una vez convertidas las direcciones lógicas a direcciones físicas mediante este programa de traducción de direcciones, ya se puede ejecutar un programa cargándolo en una posición de memoria principal.

6. Bibliografía

- Entregable S5: Memoria (dirección lógica y física). [1]
- Entregable S7: Memoria (Métodos de gestión de memoria particionamiento, segmentación y paginación). [2]
- Entregable S10: Estructuras de datos en C para cada método de gestión de memoria (particionamiento, segmentación y paginación). [3]