INFORMATICA I

Ordenamiento de un vector por el metodo burbuja

Ing.Juan Carlos Cuttitta

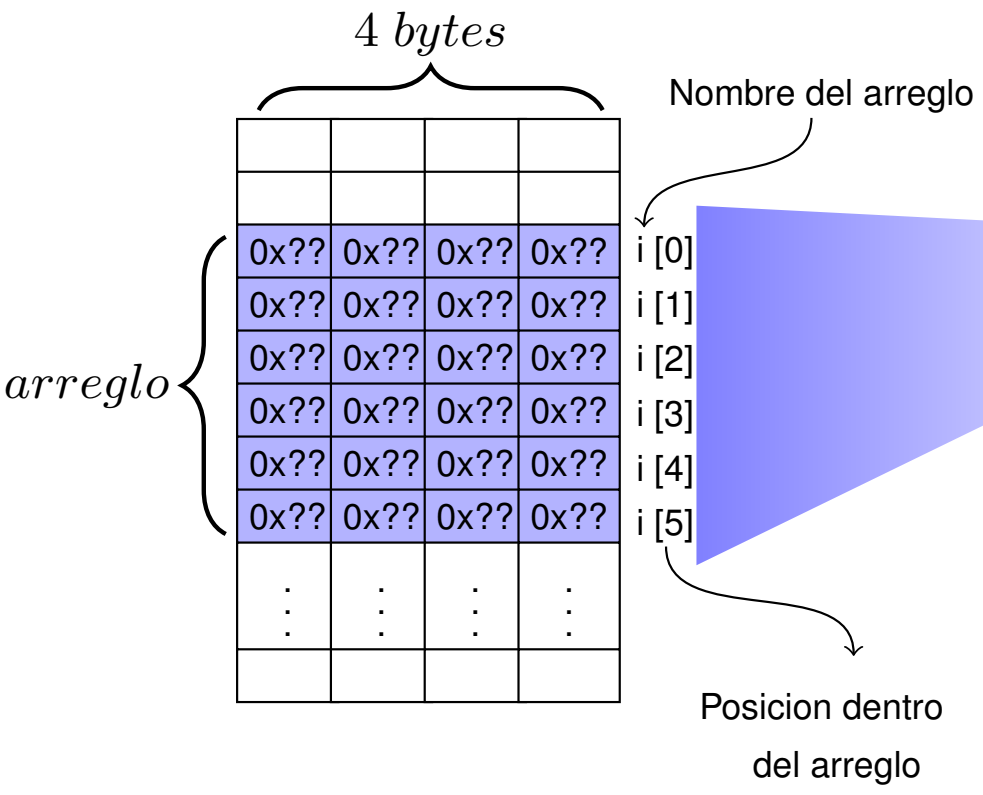
Universidad Tecnológica Nacional
Facultad Regional Buenos Aires
Departamento de Ingeniería Electrónica

4 de junio de 2020

Declaración y disposición en memoria

Arquitectura X86-32 bits

Disposición de la variable i en memoria



Código en programa fuente

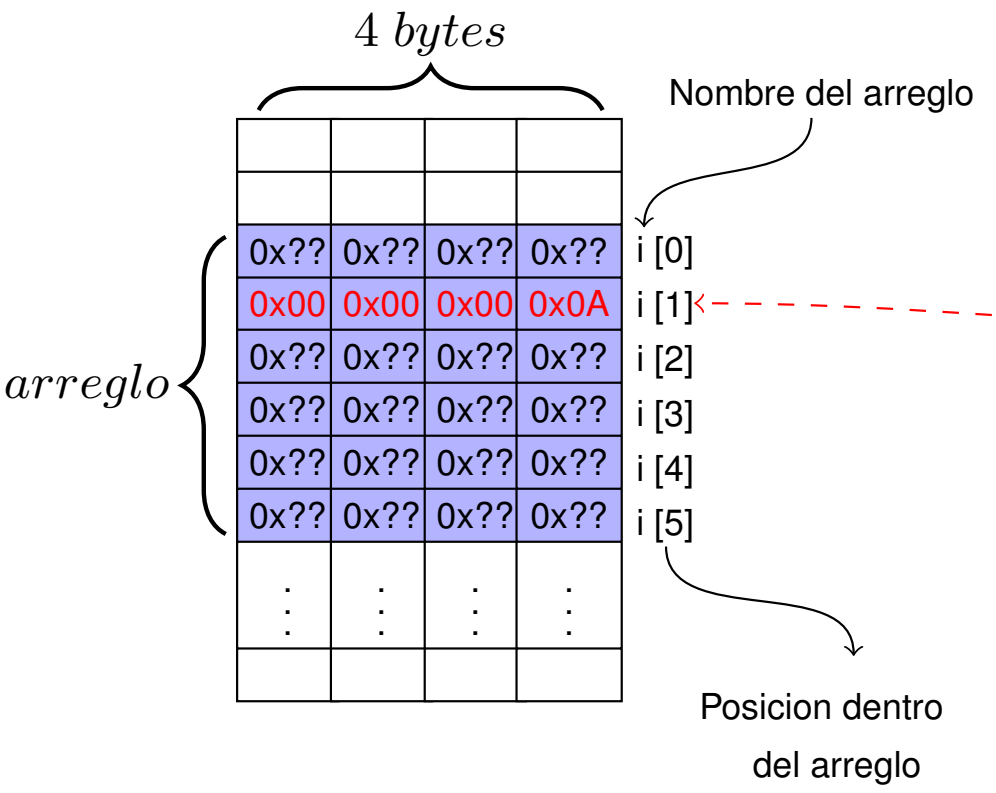
```
1   int main ()
2   {
3     int i[6];
4     .......
5     i[1]=10;
6     .......
7   }
```

Declara arreglo de 6 enteros

Acceso al contenido

Arquitectura X86-32 bits

Disposición de la variable i en memoria



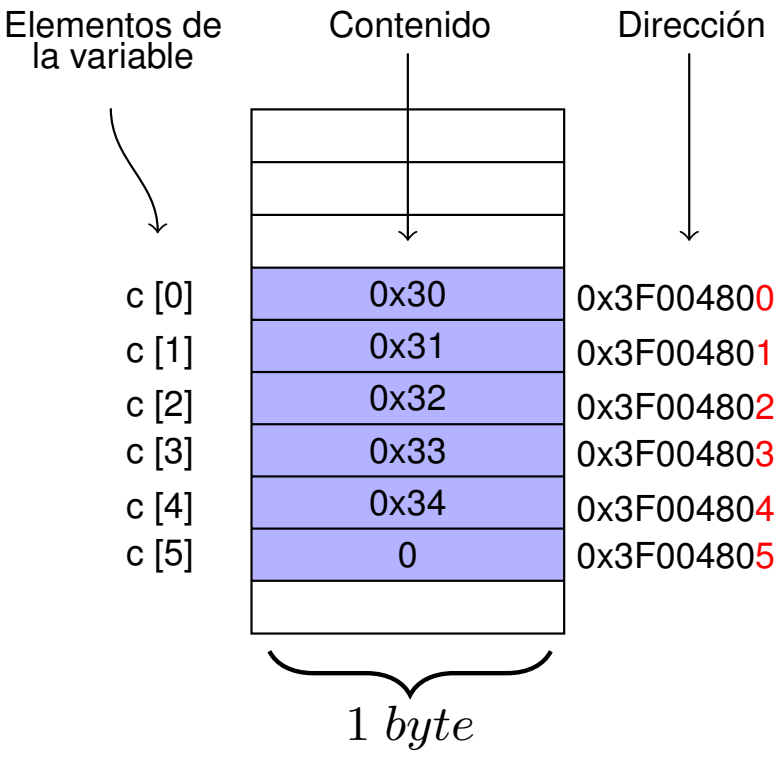
Código en programa fuente

```
1   int main ()
2   {
3     int i[6];
4     .......
5     i[1]=10;
6     .......
7   }
```

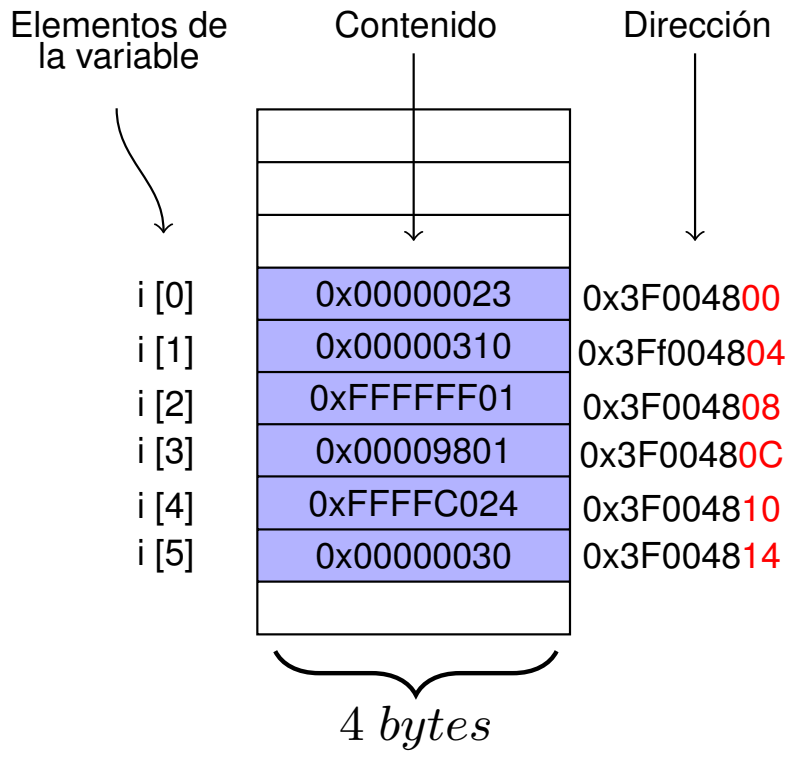
Asigna el valor 10 al elemento 1 del arreglo

Direción no es lo mismo que orden de elemento

Declaración en código
unsigned char c[6]



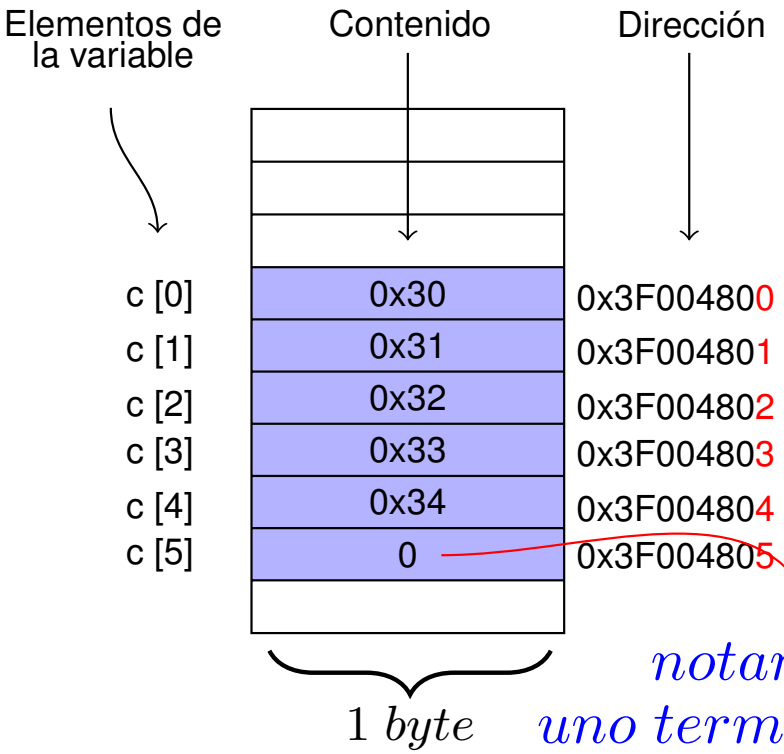
Declaración en código
int c[6]



Direción no es lo mismo que orden de elemento

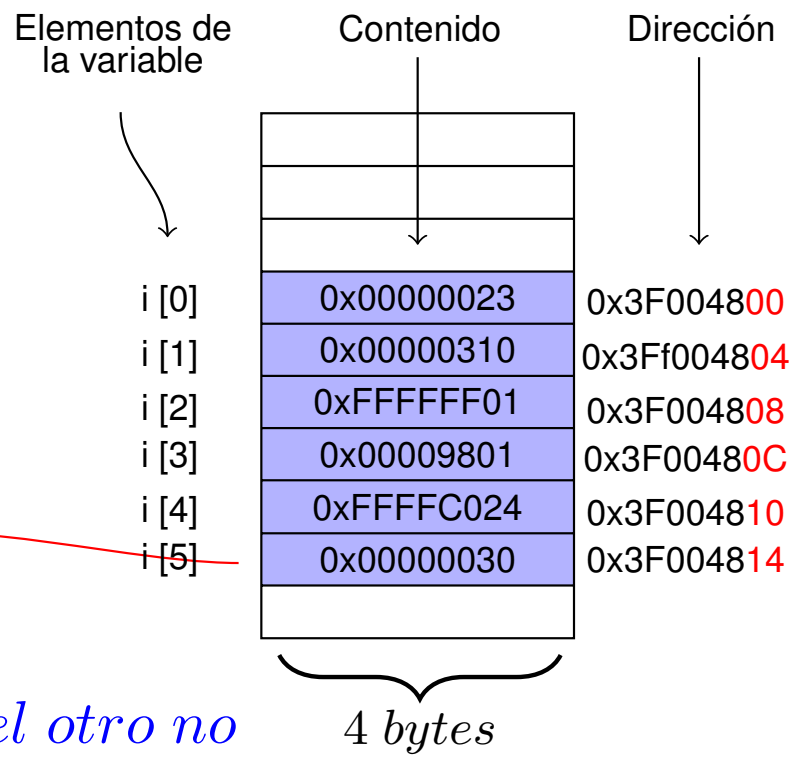
Declaración en código

unsigned char c[6]



Declaración en código

int c[6]



notar la diferencia

uno termina con '\0' y el otro no

Un caso especial de arreglo

Arreglo de caracteres

Se trata de un tipo muy usual de dato que llamamos cadena o string (del inglés).
Se inicializa de los siguientes modos:

```
1 /*Una forma de inicializar un areglo de caracteres con una
      cadena*/
2
3 char cad []= "Hola mundo!";
4
5 /* Otra forma...*/
6
7 char cad []={'H','o','l','a',' ','m','u','n','d','o','!','\0'};
```

Un caso especial de arreglo

Arreglo de caracteres

Se trata de un tipo muy usual de dato que llamamos cadena o string (del inglés).
Se inicializa de los siguientes modos:

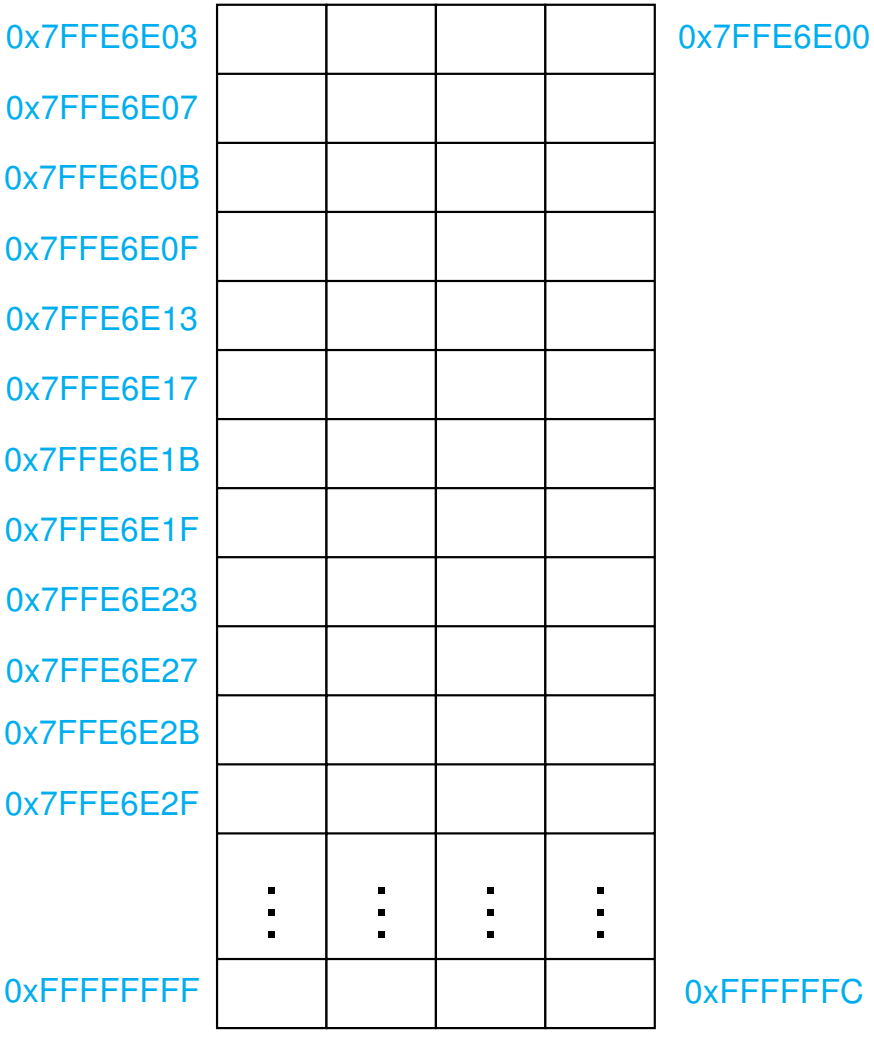
```
1 /*Una forma de inicializar un areglo de caracteres con una
      cadena*/
2
3 char cad []= "Hola mundo!";
4
5 /* Otra forma...*/
6
7 char cad []={'H','o','l','a',' ','m','u','n','d','o','!','\0'};
```

En código....

```
1   /* Utiliza una lista de inicialización para
       inicializar el arreglo arr.*/
2   int arr[8] = {23,0,-669,-1,995,1277,90,-9000};
3   /* La cantidad de elementos en la línea anterior
       es redundante. Puede hacerse lo mismo de la
       siguiente forma*/
4   int arr[] = {23,0,-669,-1,995,1277,90,-9000};
5   /*Si lo vamos a inicializar con el mismo valor
       para todos los elementos, la forma adecuada
       es la siguiente*/
6
7   int arr[8], i;
8   for (i = 0 ; i < 8 ; i++)
9   {
10    arr[i] = 0;
11  }
```

Ordenamiento de un vector por el metodo ***burbuja***

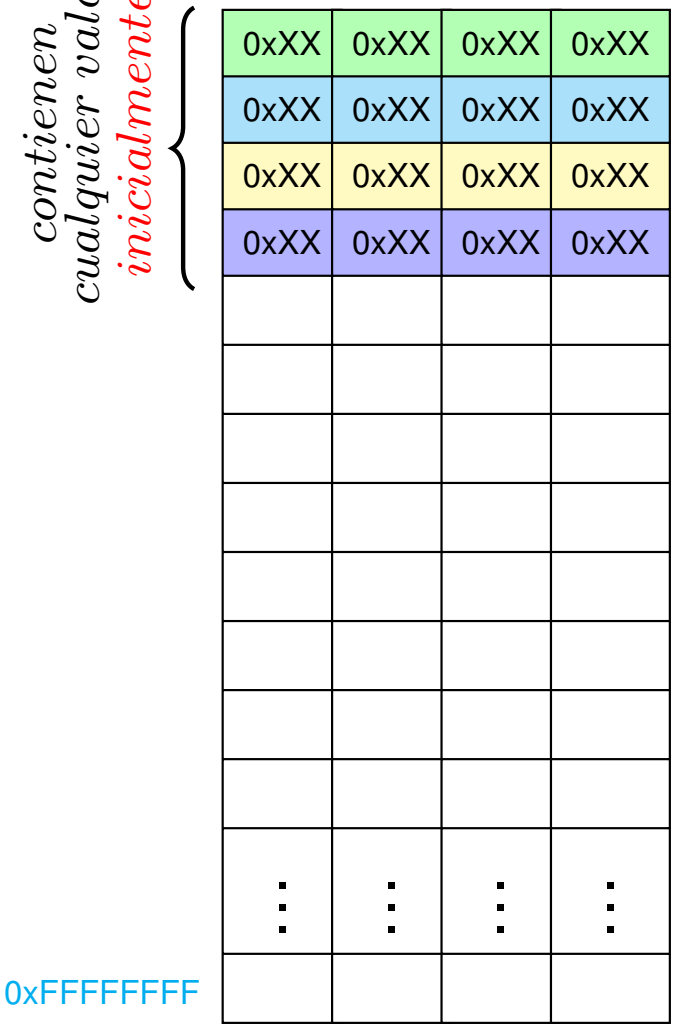
MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

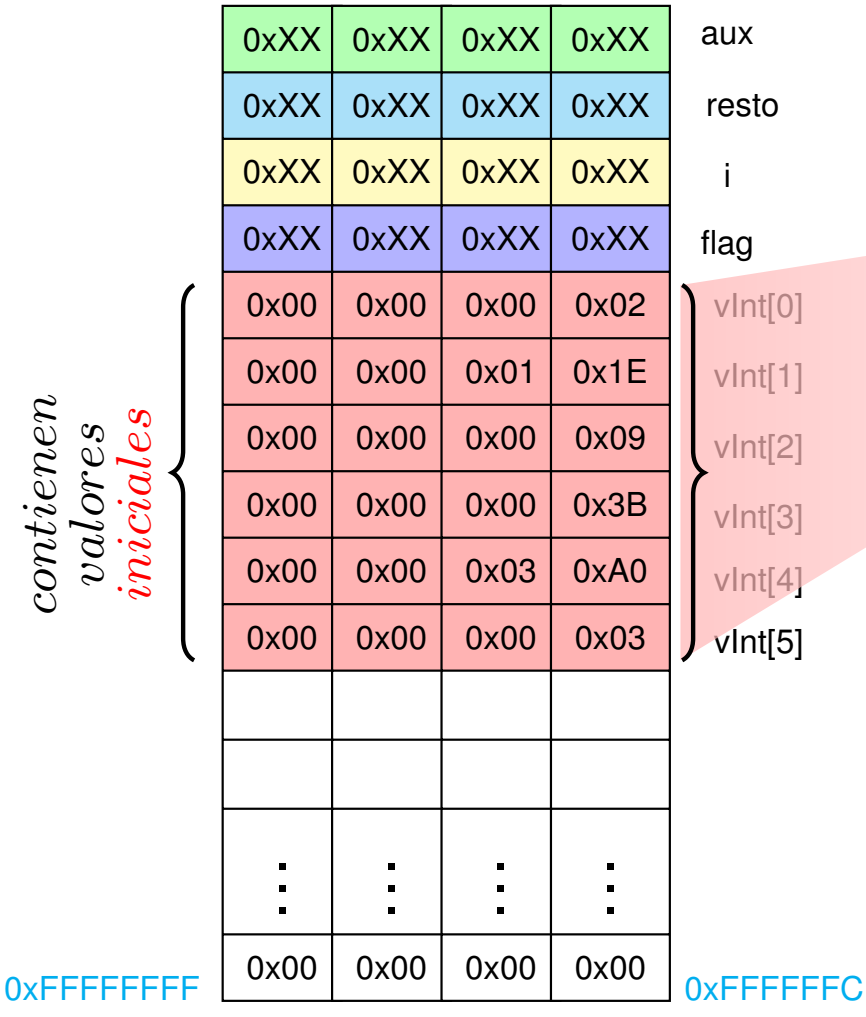
MEMORIA en ARQUITECTURA x86 32-bits

contienen cualquier valor inicialmente



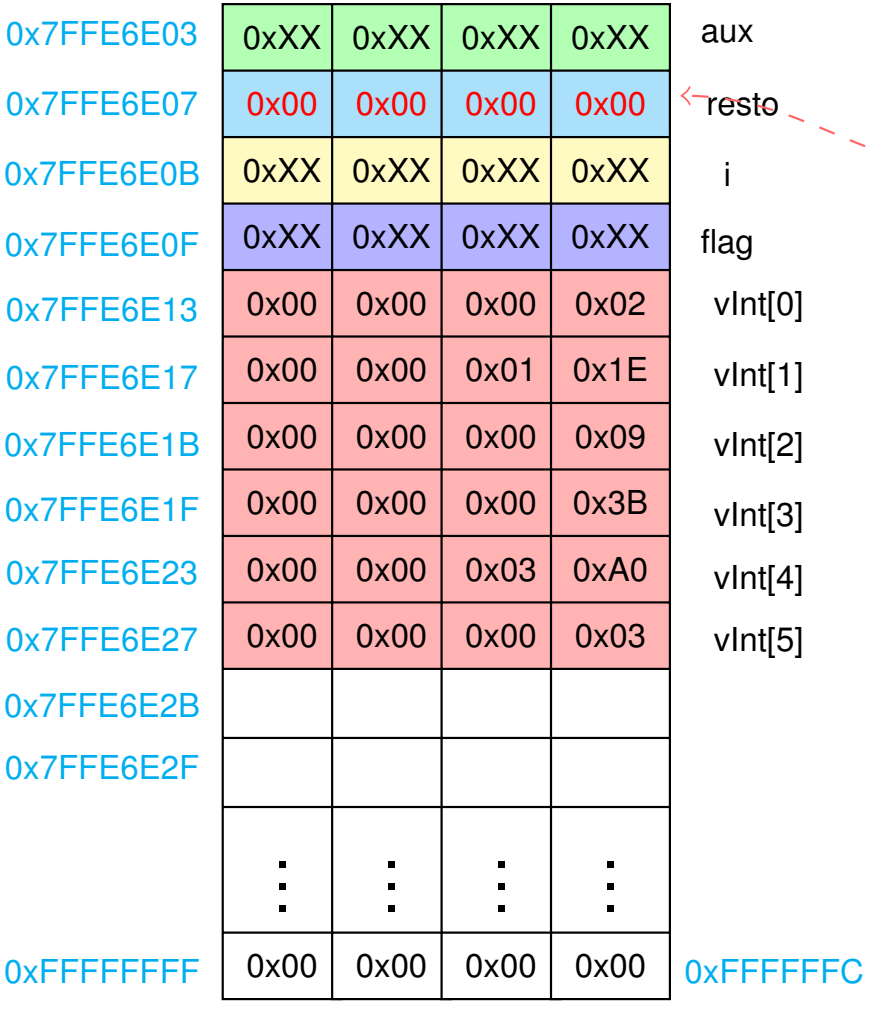

```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

MEMORIA en ARQUITECTURA x86 32-bits



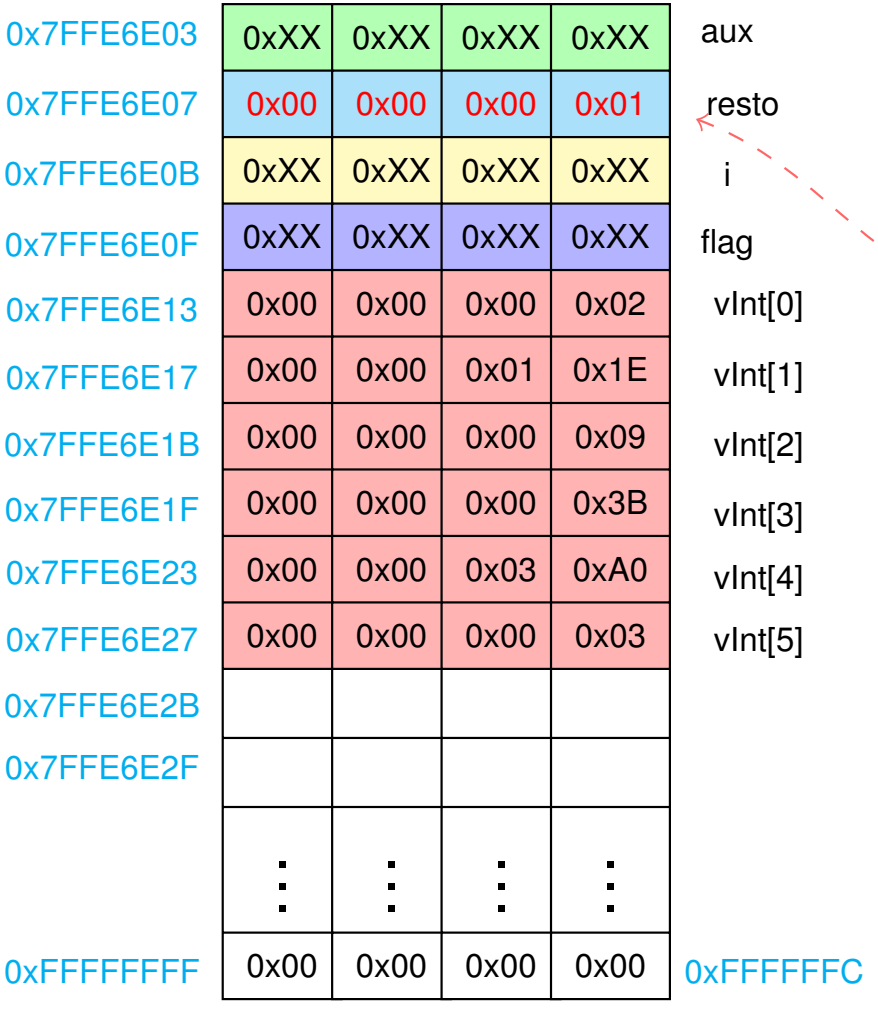

```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

MEMORIA en ARQUITECTURA x86 32-bits



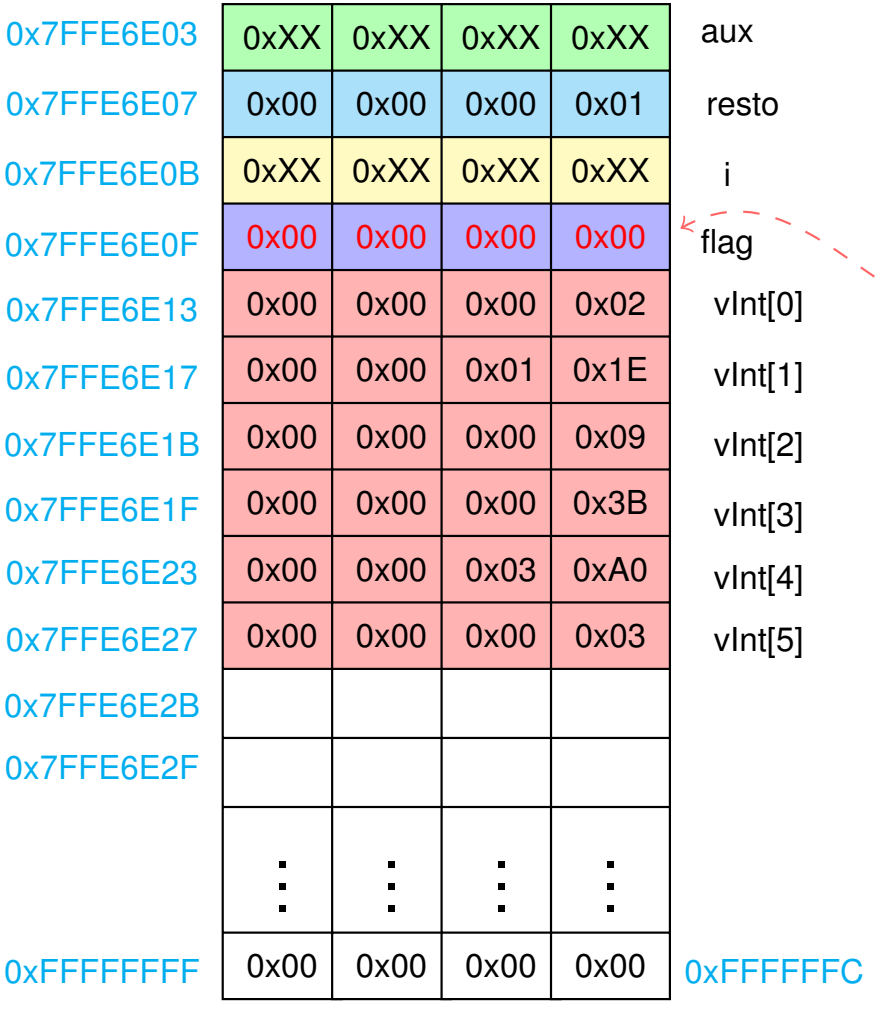

```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

MEMORIA en ARQUITECTURA x86 32-bits



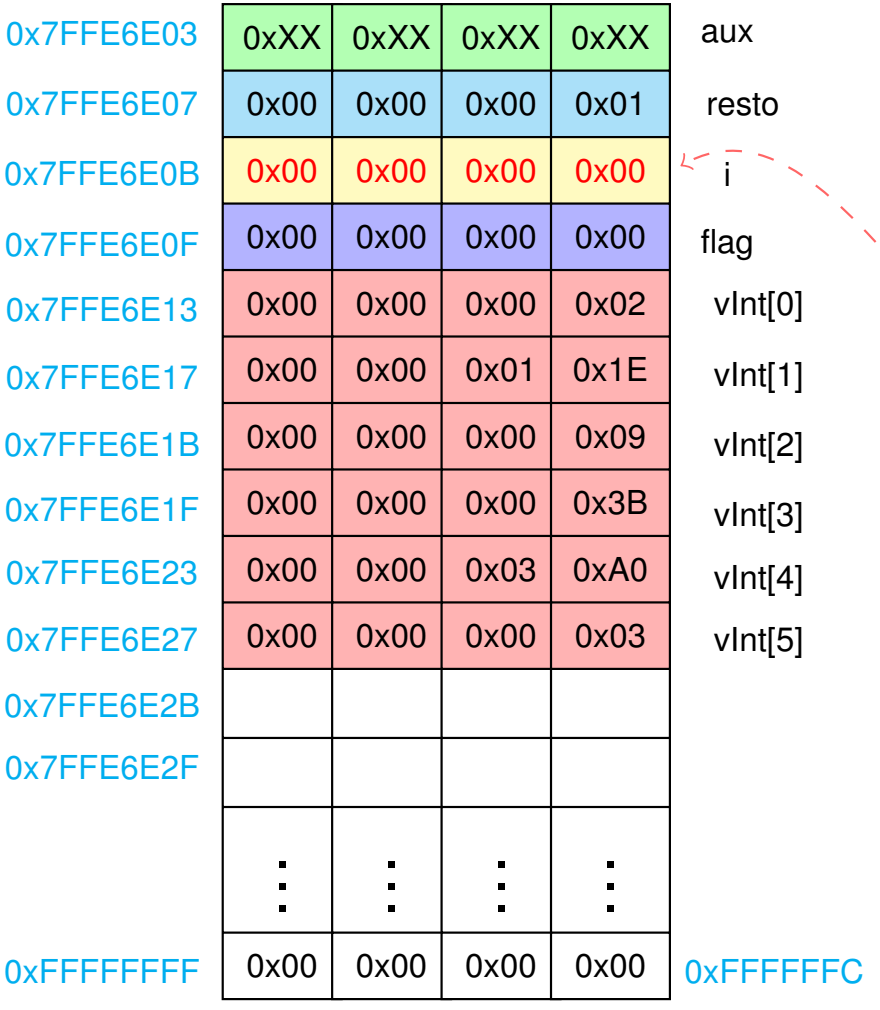

```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

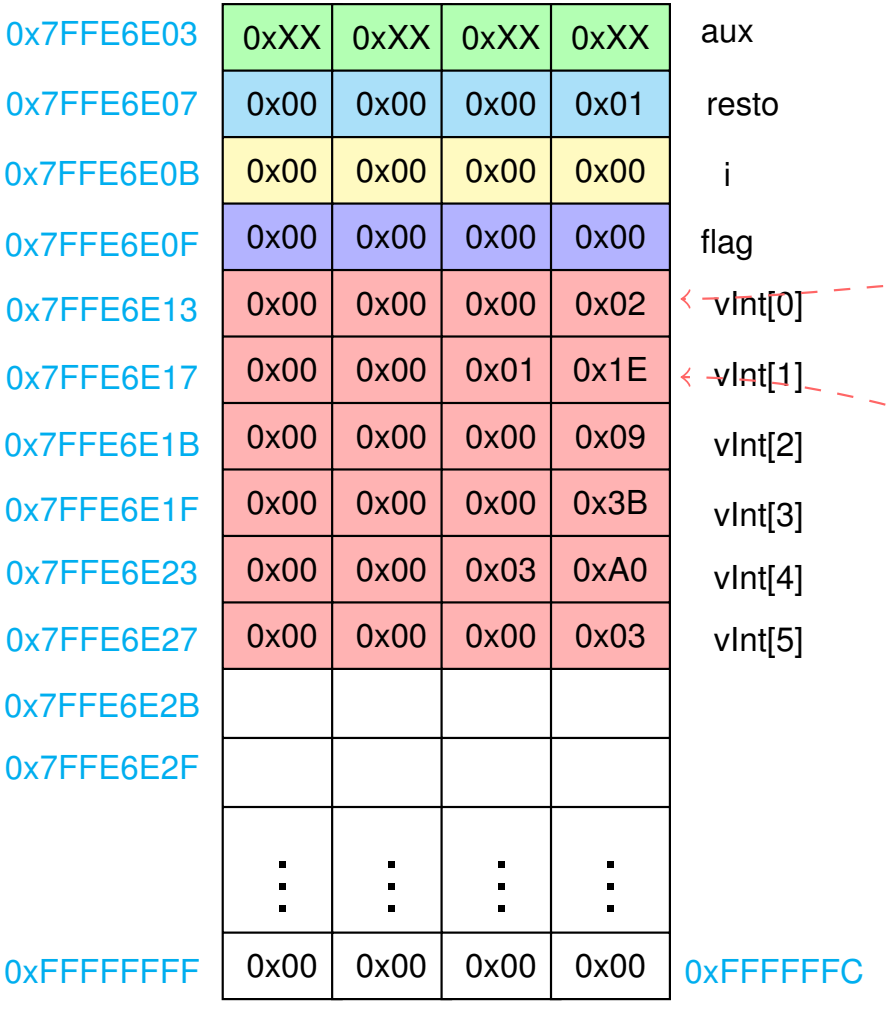
MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-1=5

MEMORIA en ARQUITECTURA x86 32-bits

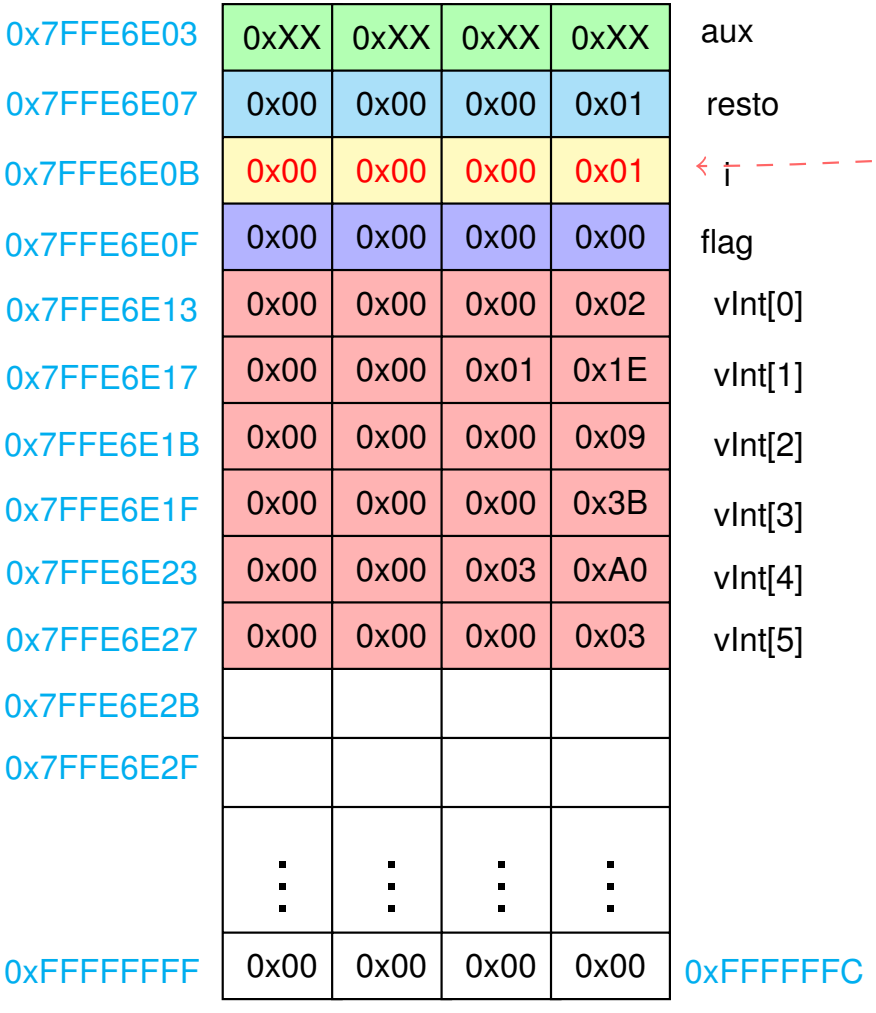



```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-1=5

0 1

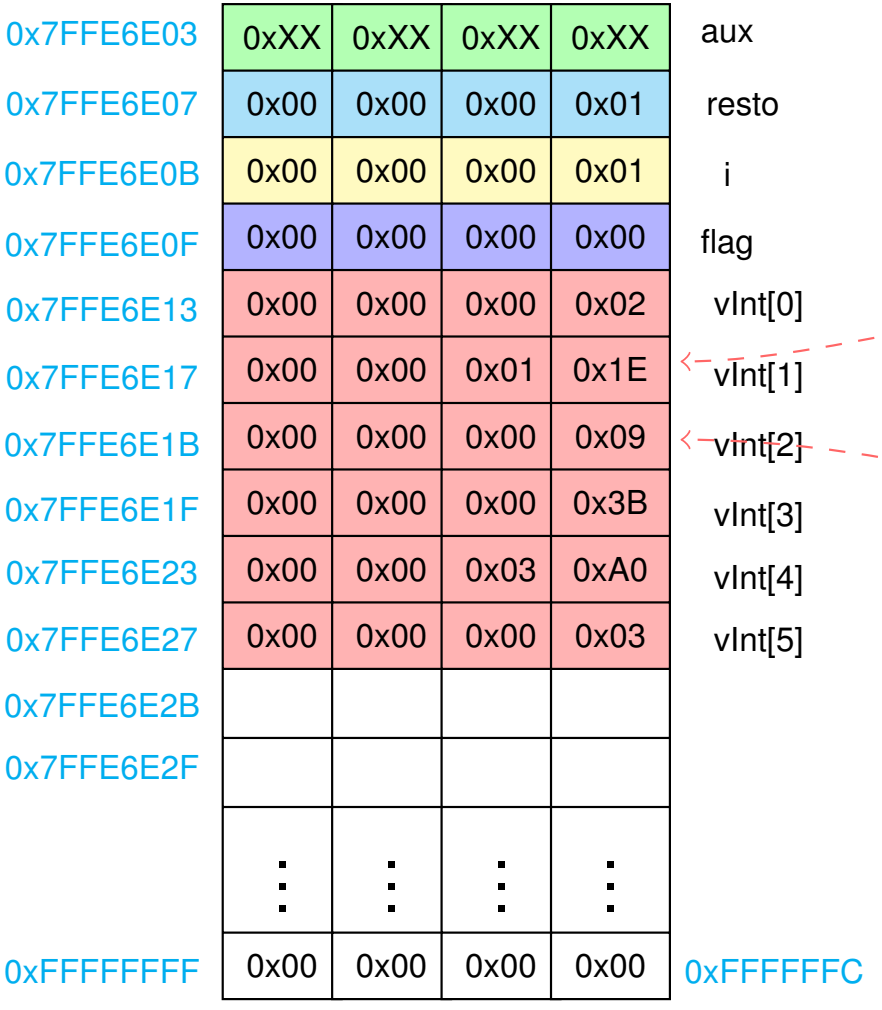
MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-1=5

MEMORIA en ARQUITECTURA x86 32-bits

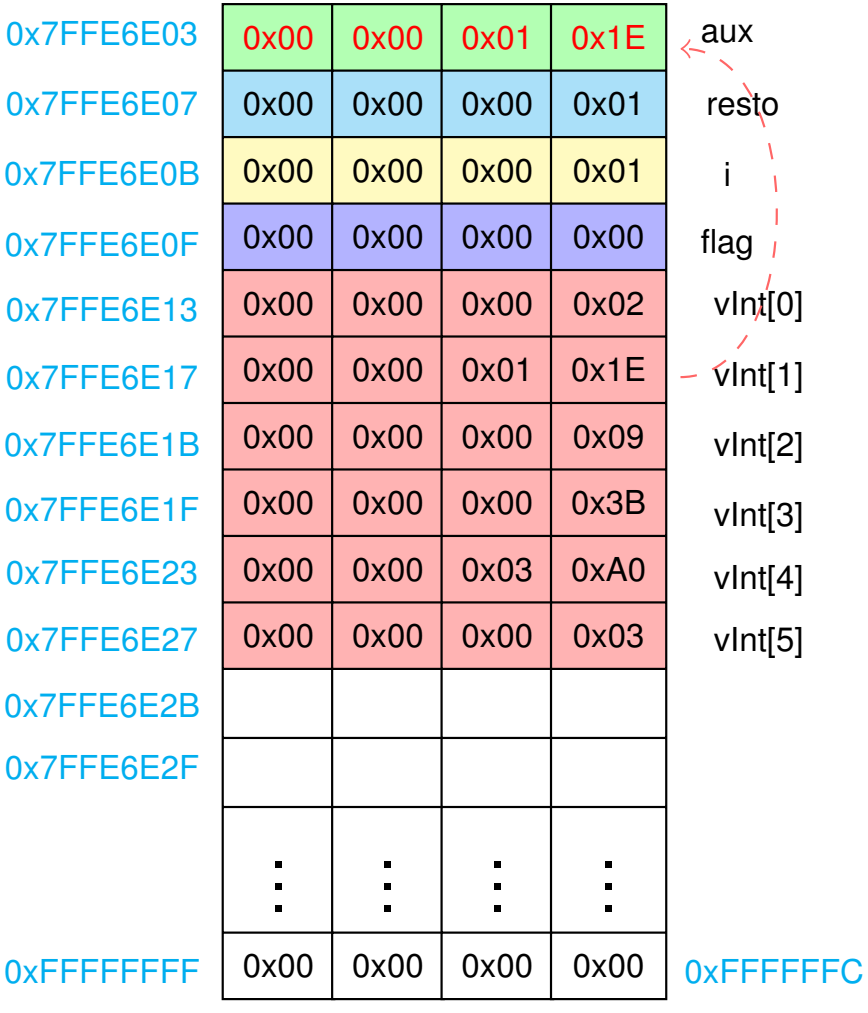



```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-1=5

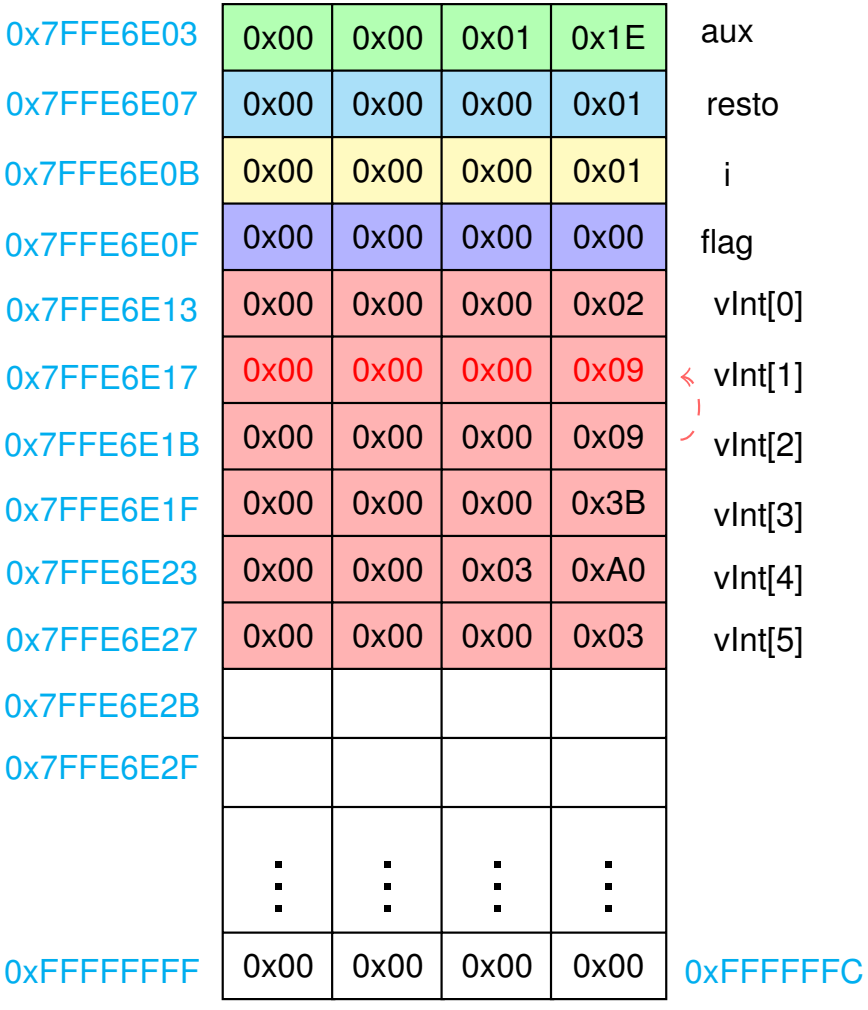
1 2

MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;                    6-1=5
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {                  1
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

MEMORIA en ARQUITECTURA x86 32-bits

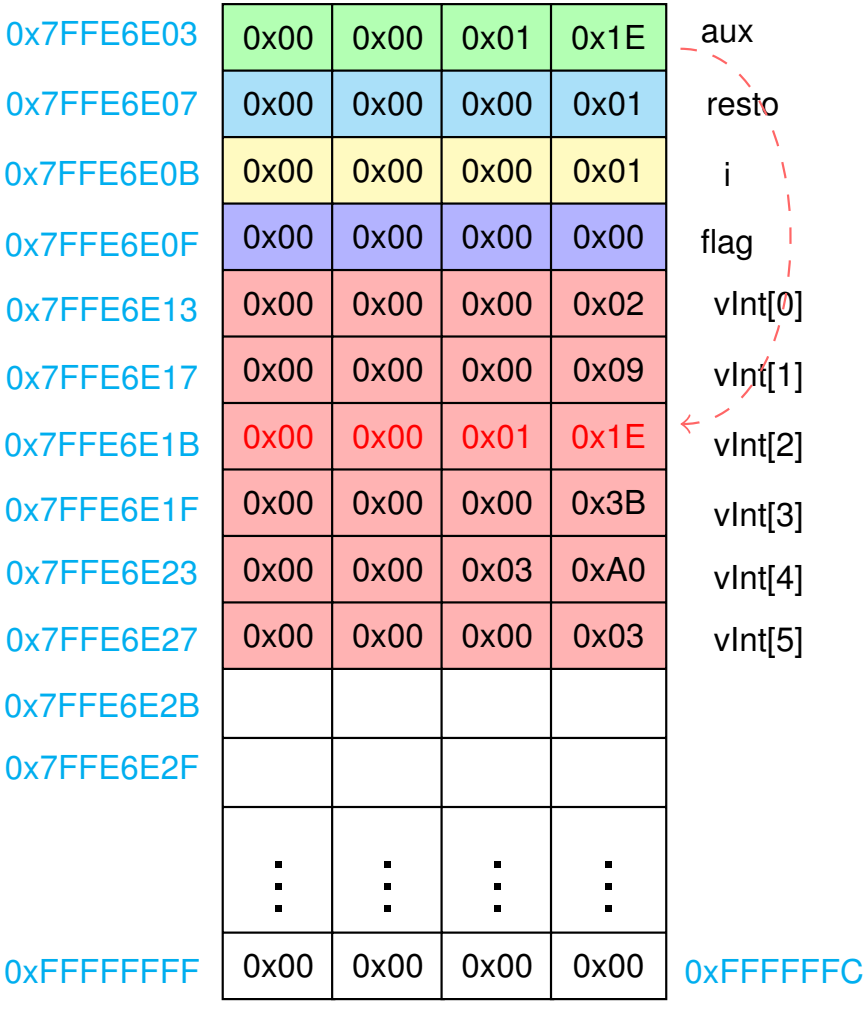



```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-1=5

vInt[1] = vInt[2]

MEMORIA en ARQUITECTURA x86 32-bits

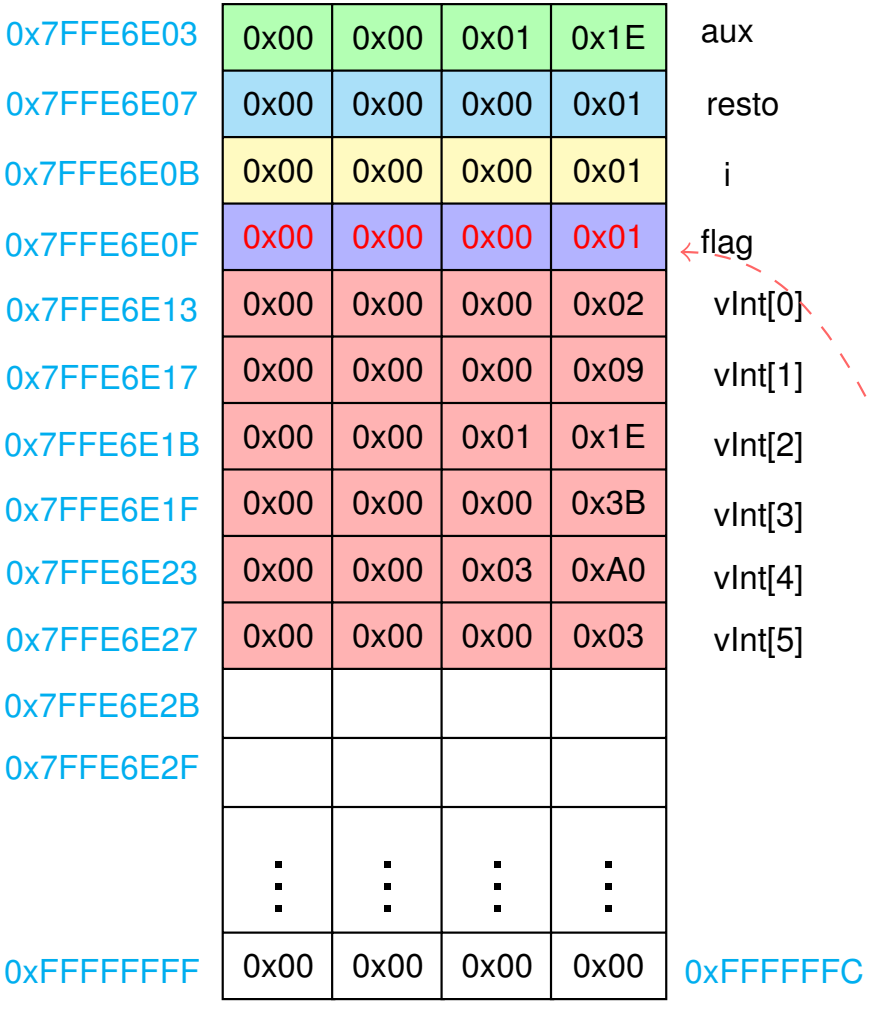



```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-1=5

vInt[2] = aux

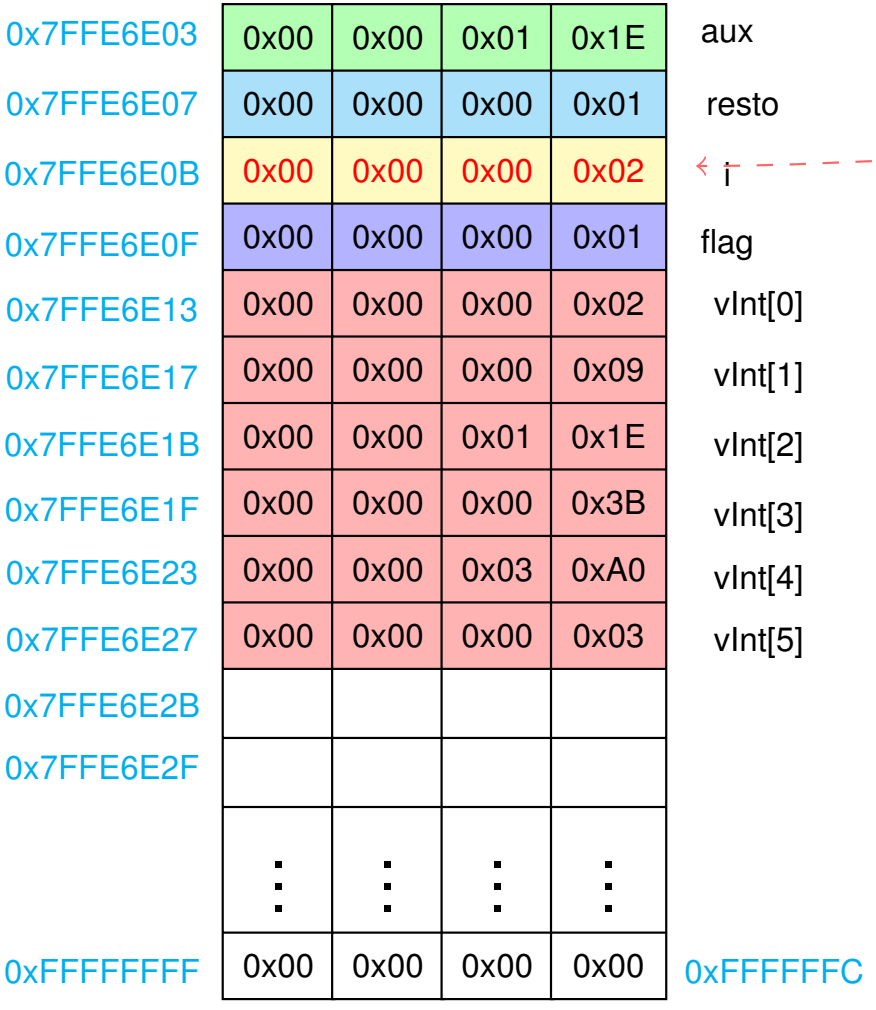
MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-1=5

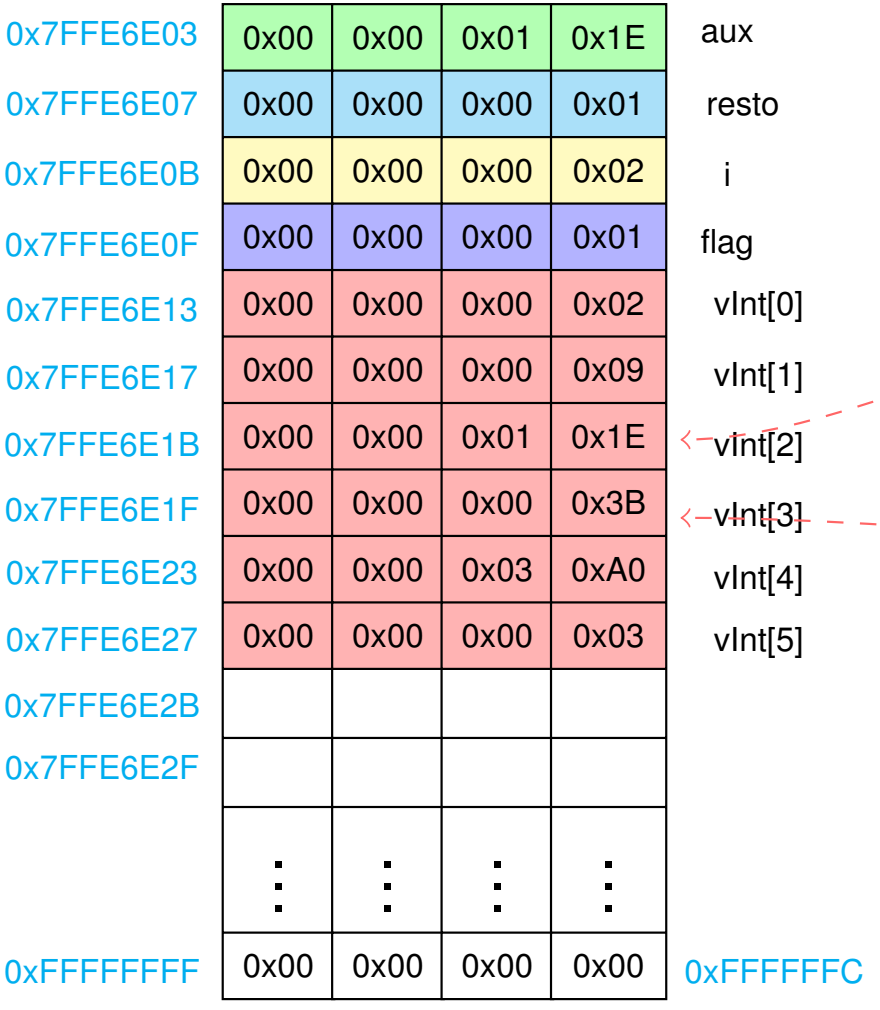
MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

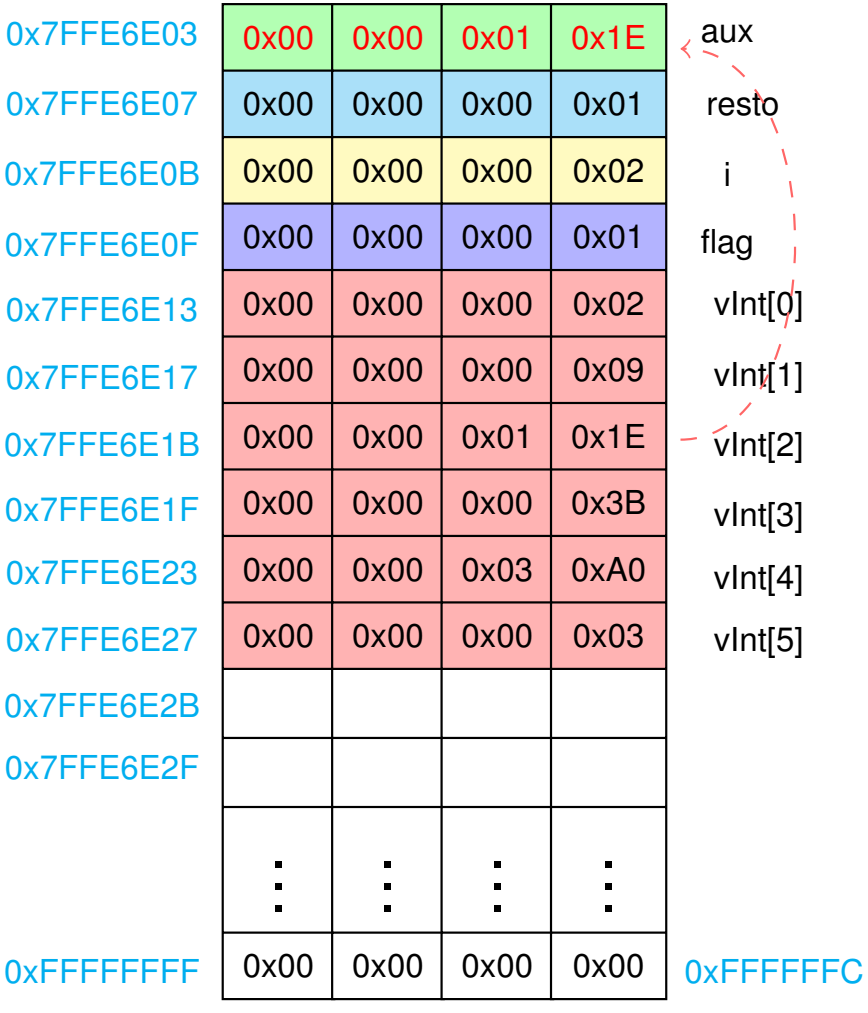
6-1=5

MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;                    6-1=5
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {          2                3
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-1=5

2

MEMORIA en ARQUITECTURA x86 32-bits

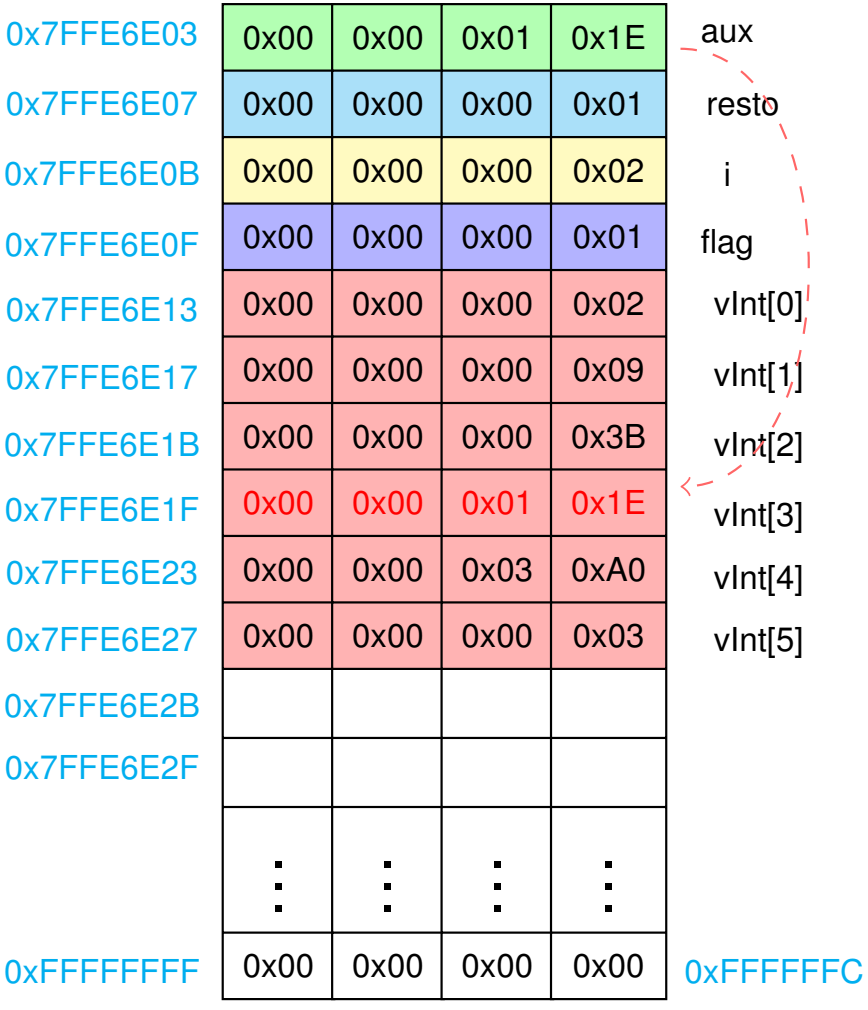
0x7FFE6E03	0x00	0x00	0x01	0x1E	aux
0x7FFE6E07	0x00	0x00	0x00	0x01	resto
0x7FFE6E0B	0x00	0x00	0x00	0x02	i
0x7FFE6E0F	0x00	0x00	0x00	0x01	flag
0x7FFE6E13	0x00	0x00	0x00	0x02	vInt[0]
0x7FFE6E17	0x00	0x00	0x00	0x09	vInt[1]
0x7FFE6E1B	0x00	0x00	0x00	0x3B	vInt[2]
0x7FFE6E1F	0x00	0x00	0x00	0x3B	vInt[3]
0x7FFE6E23	0x00	0x00	0x03	0xA0	vInt[4]
0x7FFE6E27	0x00	0x00	0x00	0x03	vInt[5]
0x7FFE6E2B					
0x7FFE6E2F					
	⋮	⋮	⋮	⋮	
0xFFFFFFFF	0x00	0x00	0x00	0x00	0xFFFFFFC

```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-1=5

vInt[2] = vInt[3]

MEMORIA en ARQUITECTURA x86 32-bits

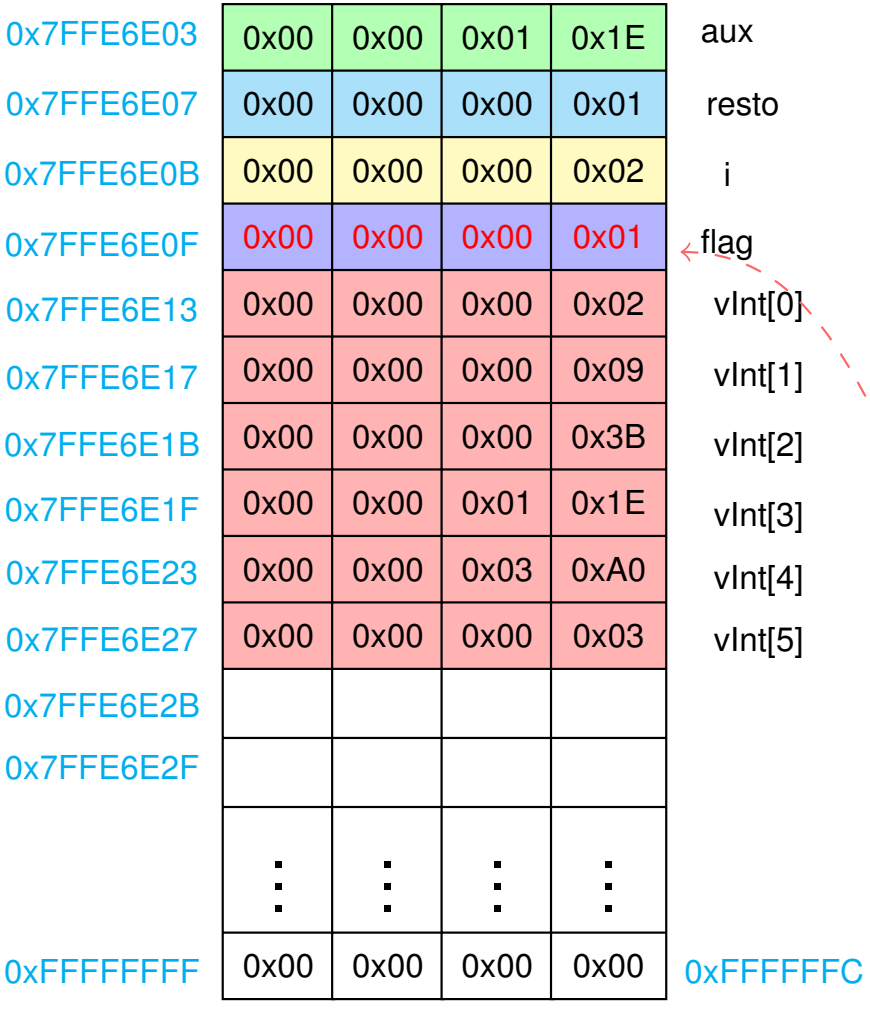



```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-1=5

vInt[3] = aux

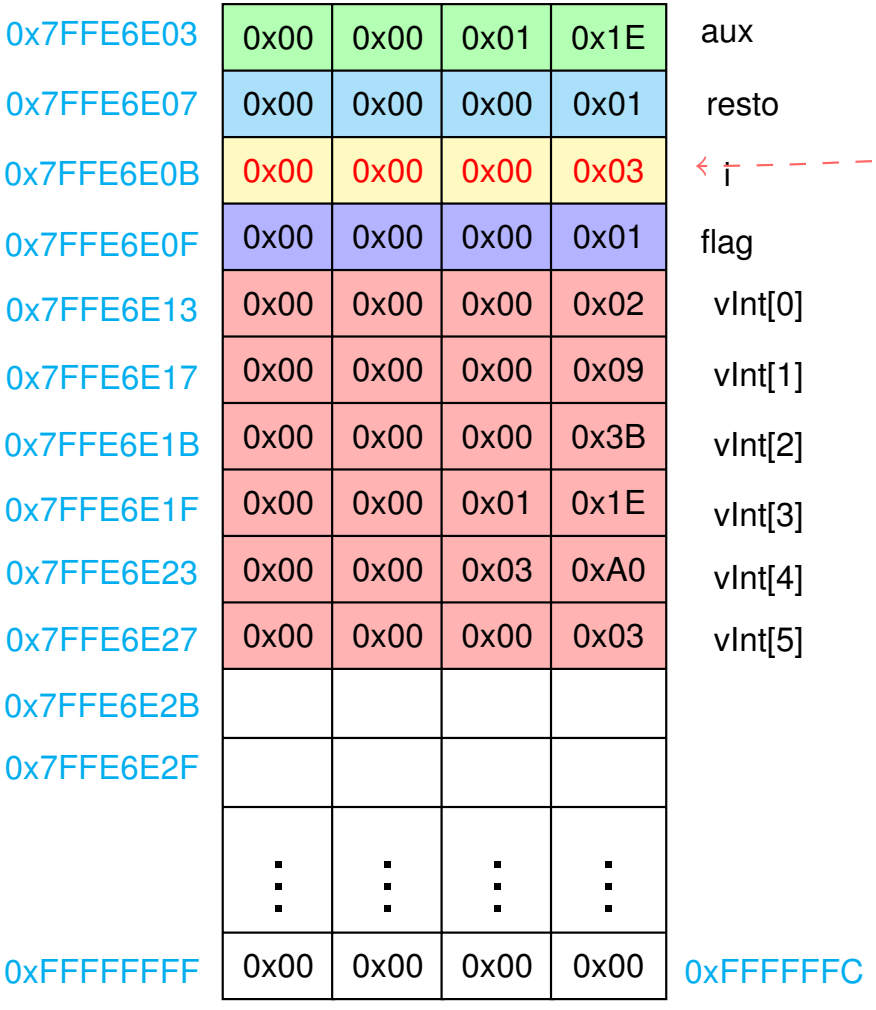
MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-1=5

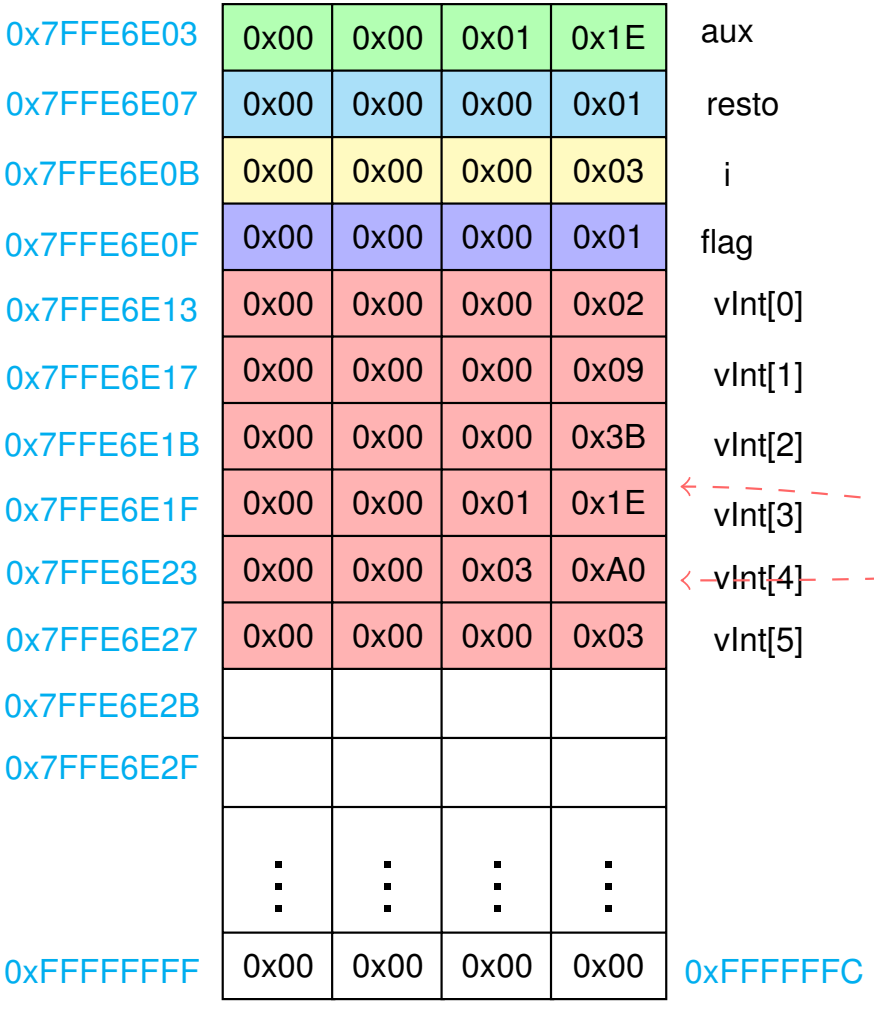
MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

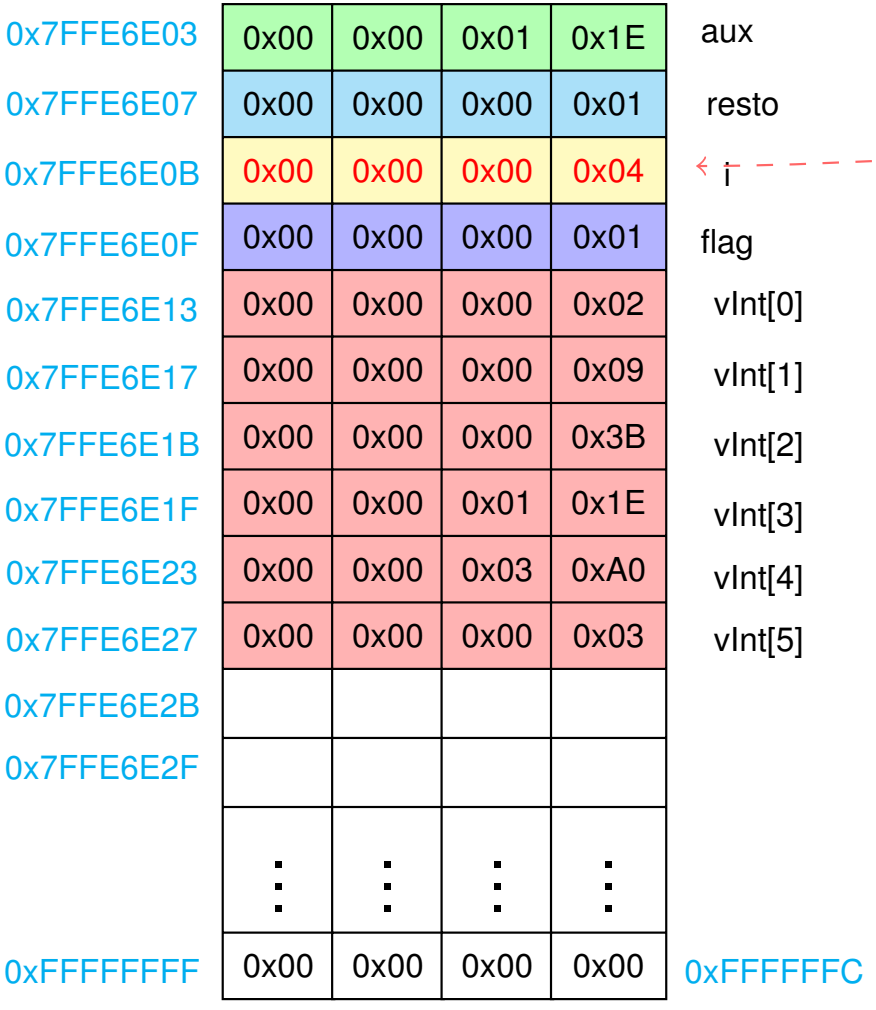
6-1=5

MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;                    6-1=5
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {                  3              4
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

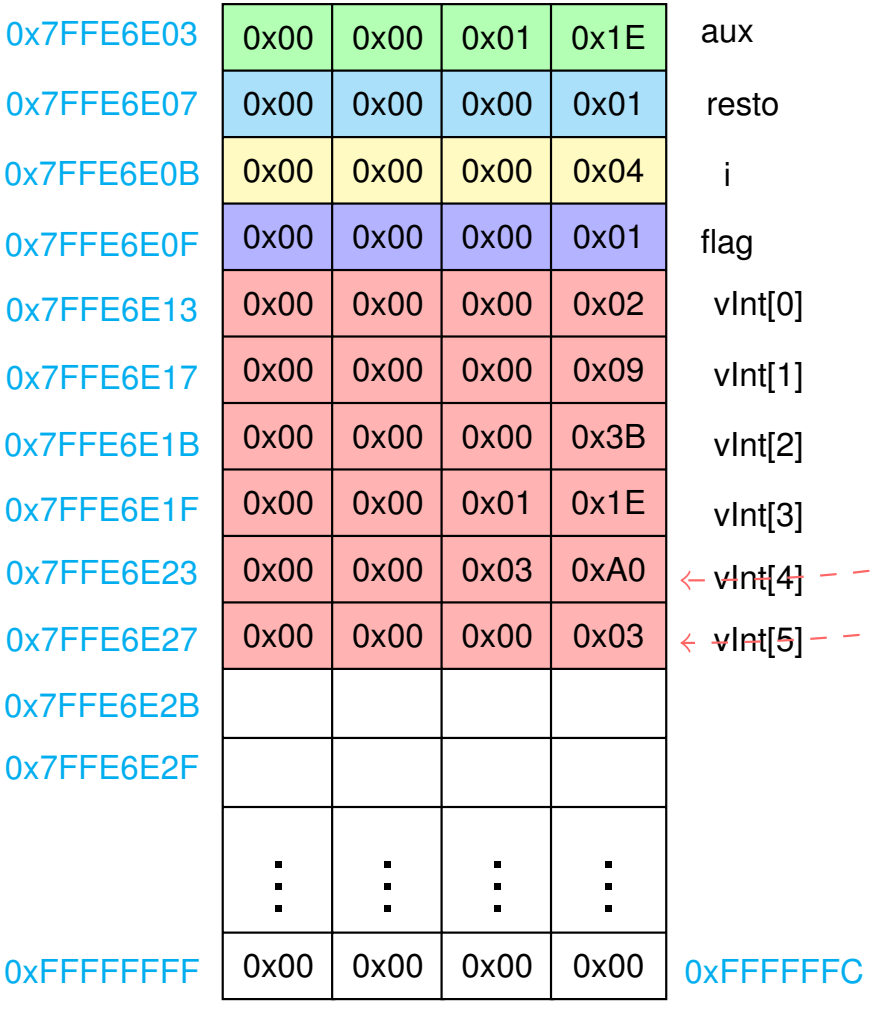
MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

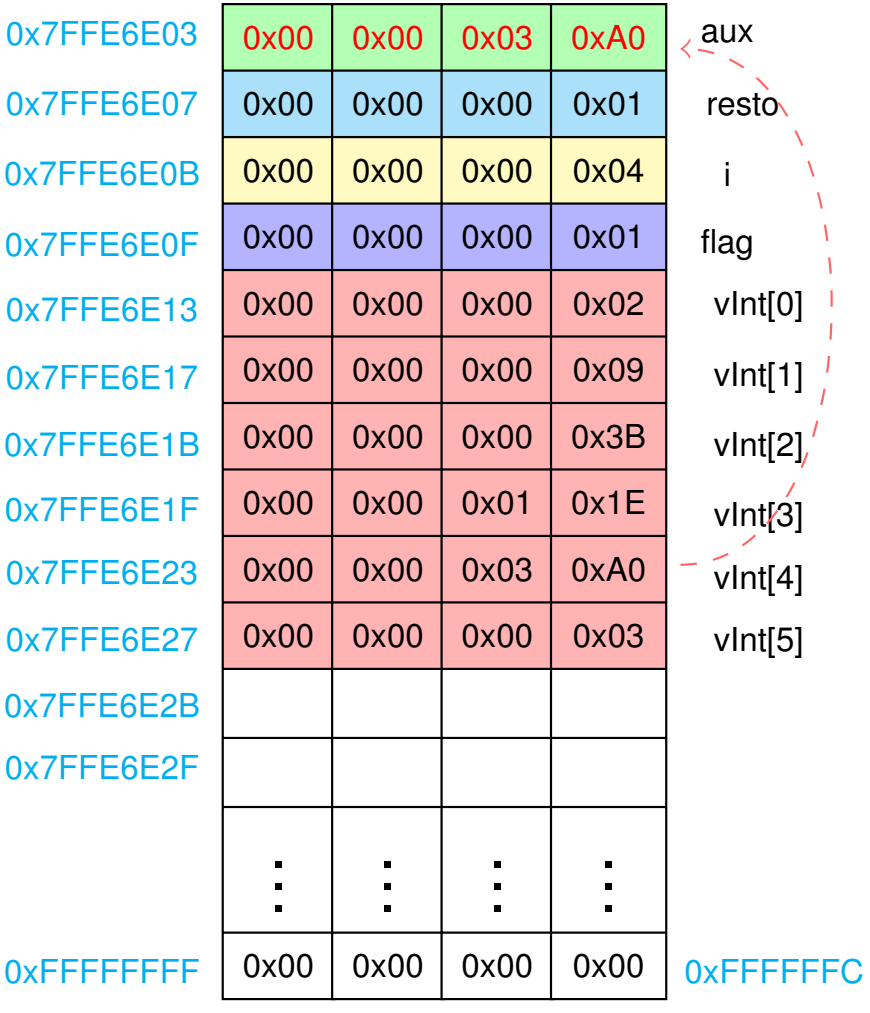
6-1=5

MEMORIA en ARQUITECTURA x86 32-bits



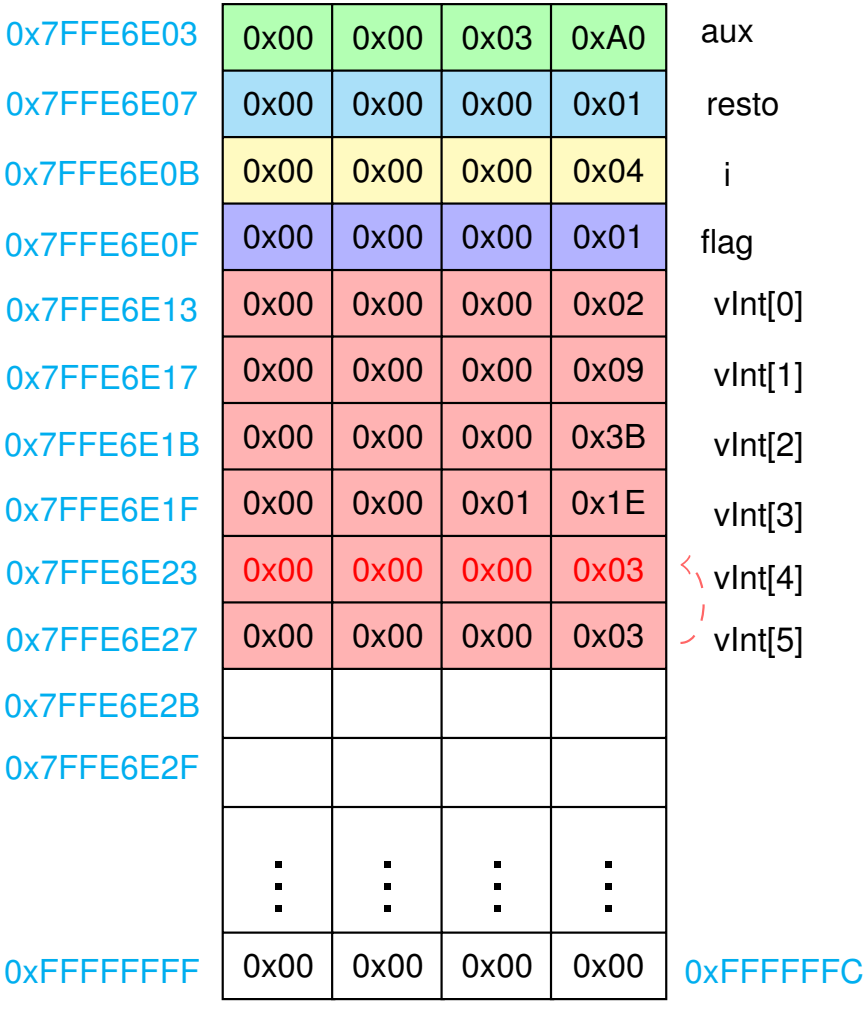

```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;          6-1=5
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {        4              5
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;          6-1=5
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {          4
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

MEMORIA en ARQUITECTURA x86 32-bits

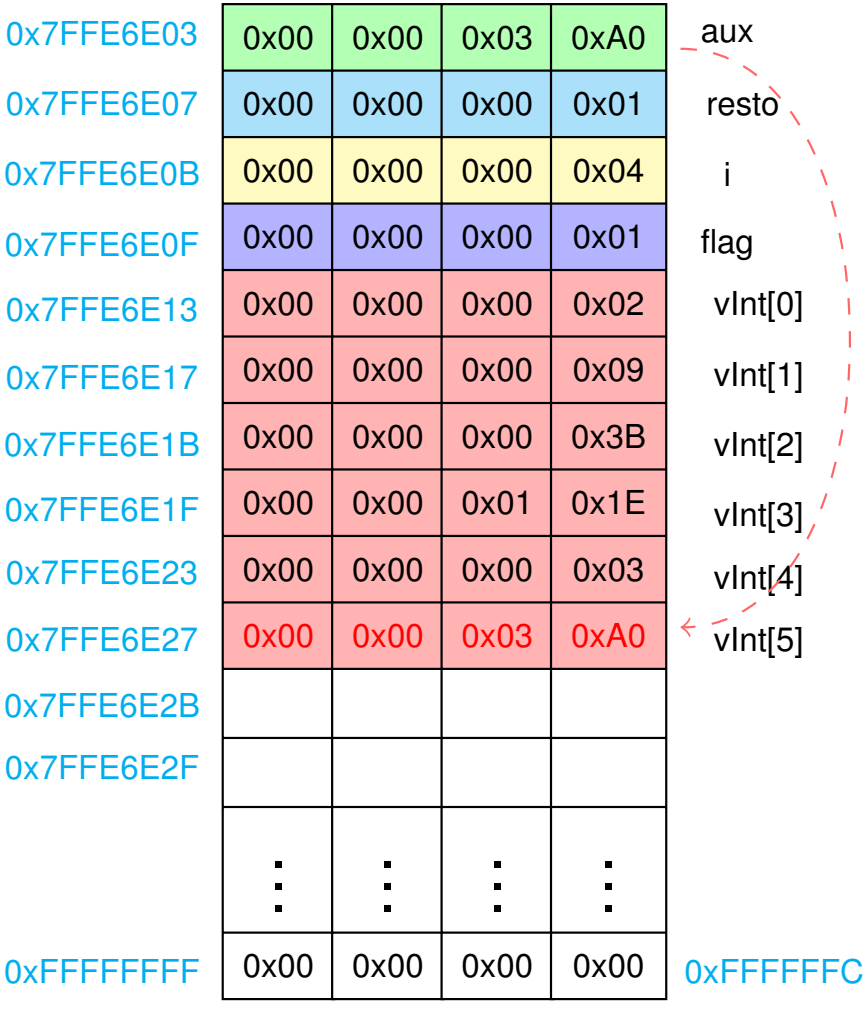



```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-1=5

vInt[4] = vInt[5]

MEMORIA en ARQUITECTURA x86 32-bits

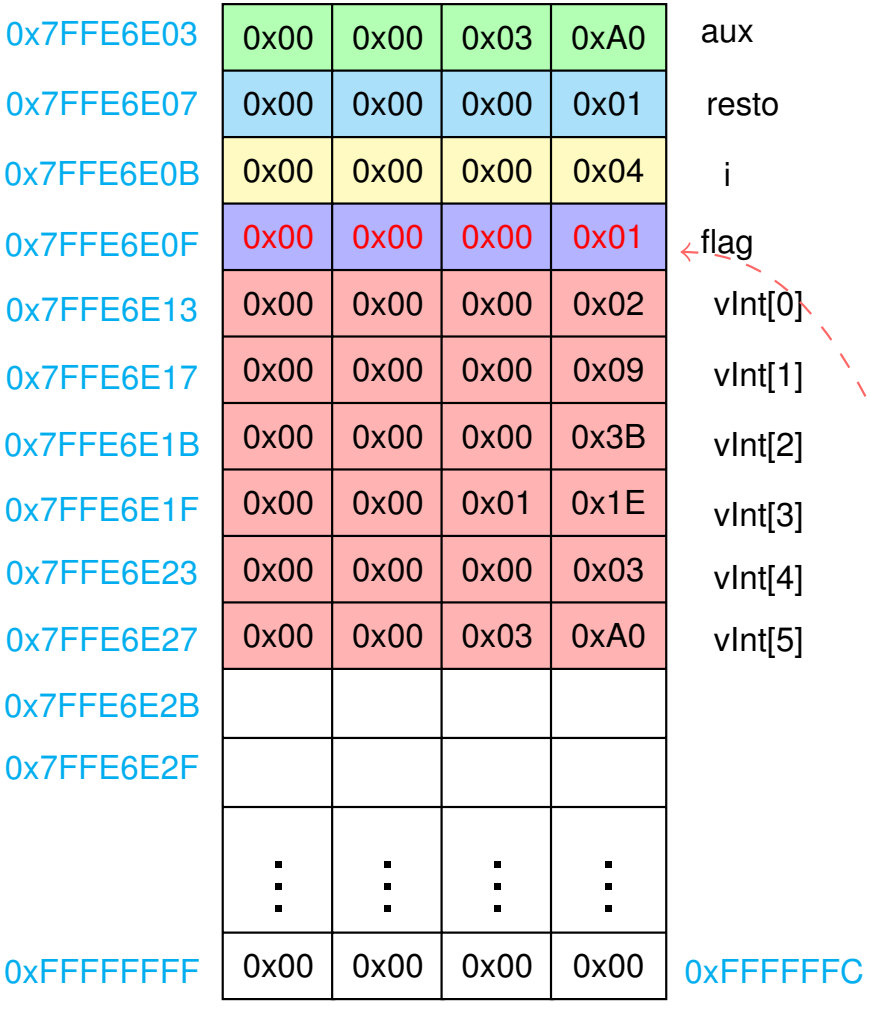



```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-1=5

vInt[5] = aux

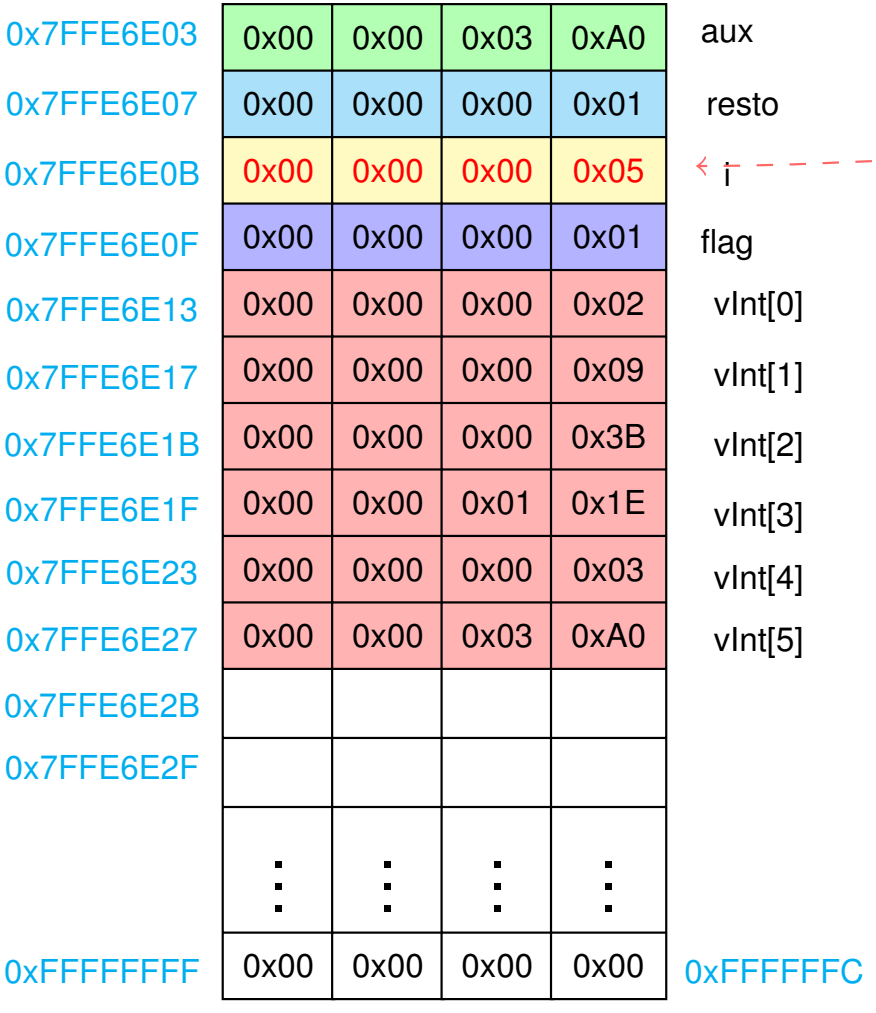
MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-1=5

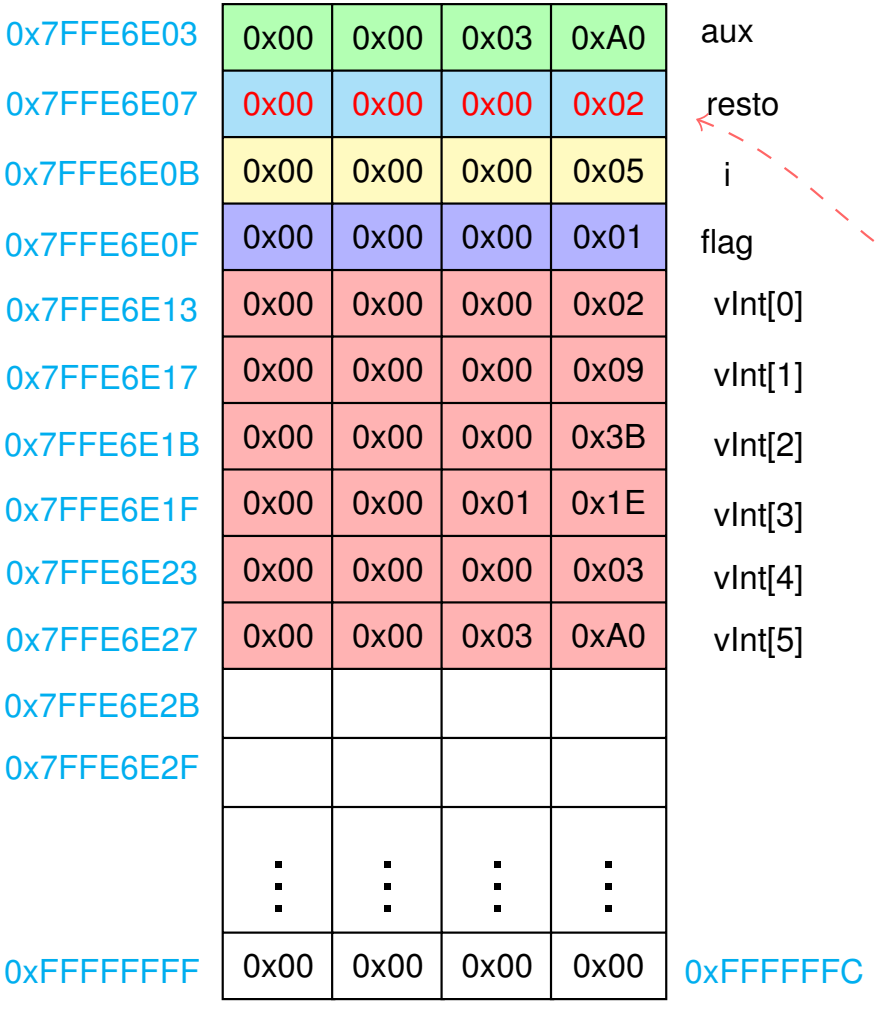
MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-1=5

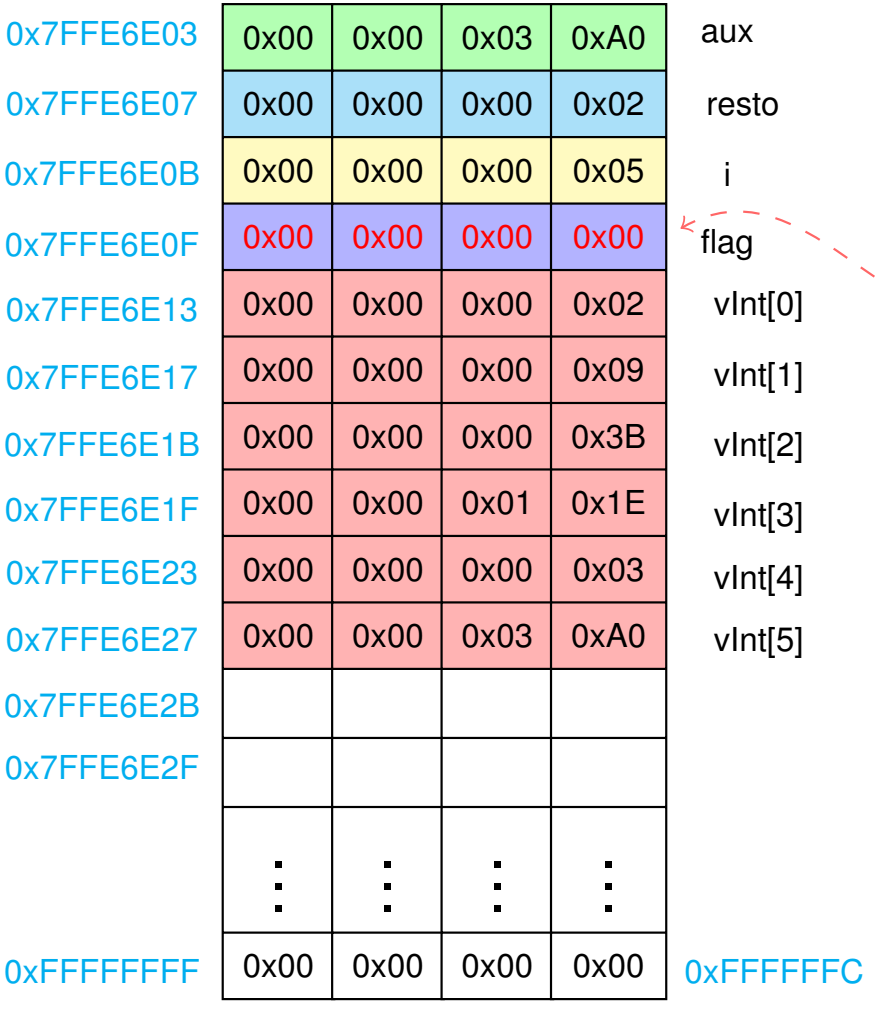
MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-1=5

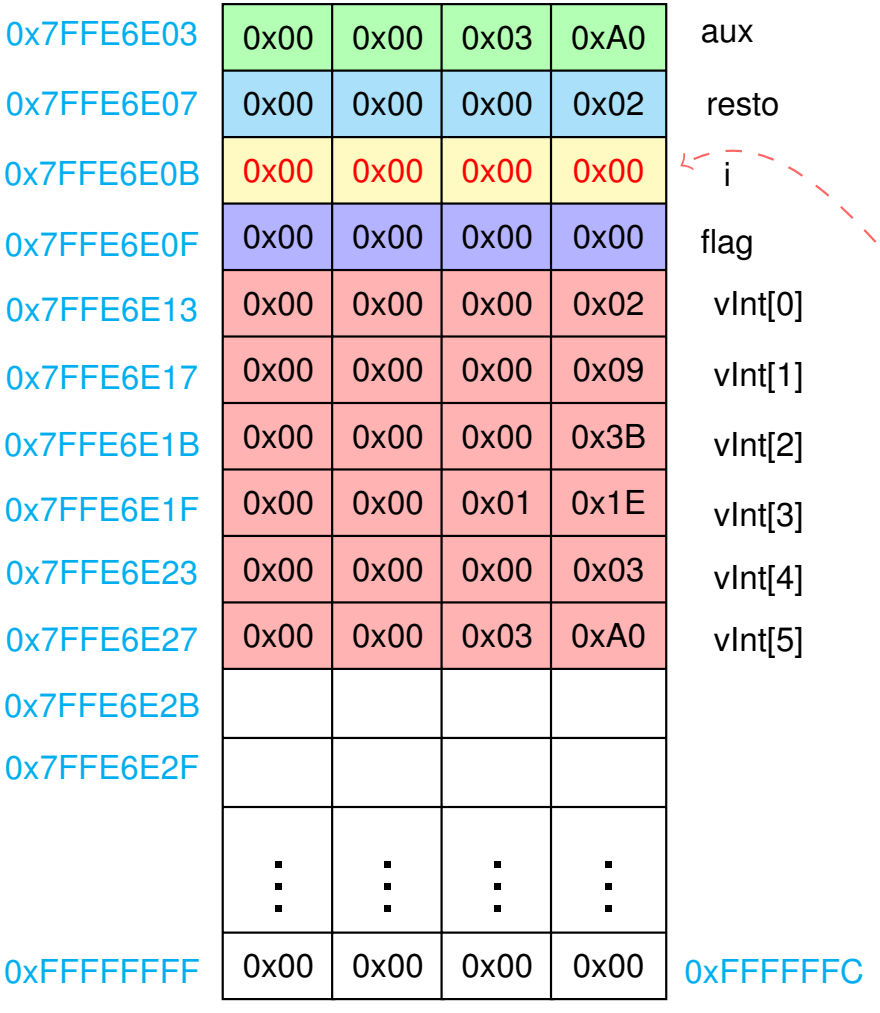
MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-1=5

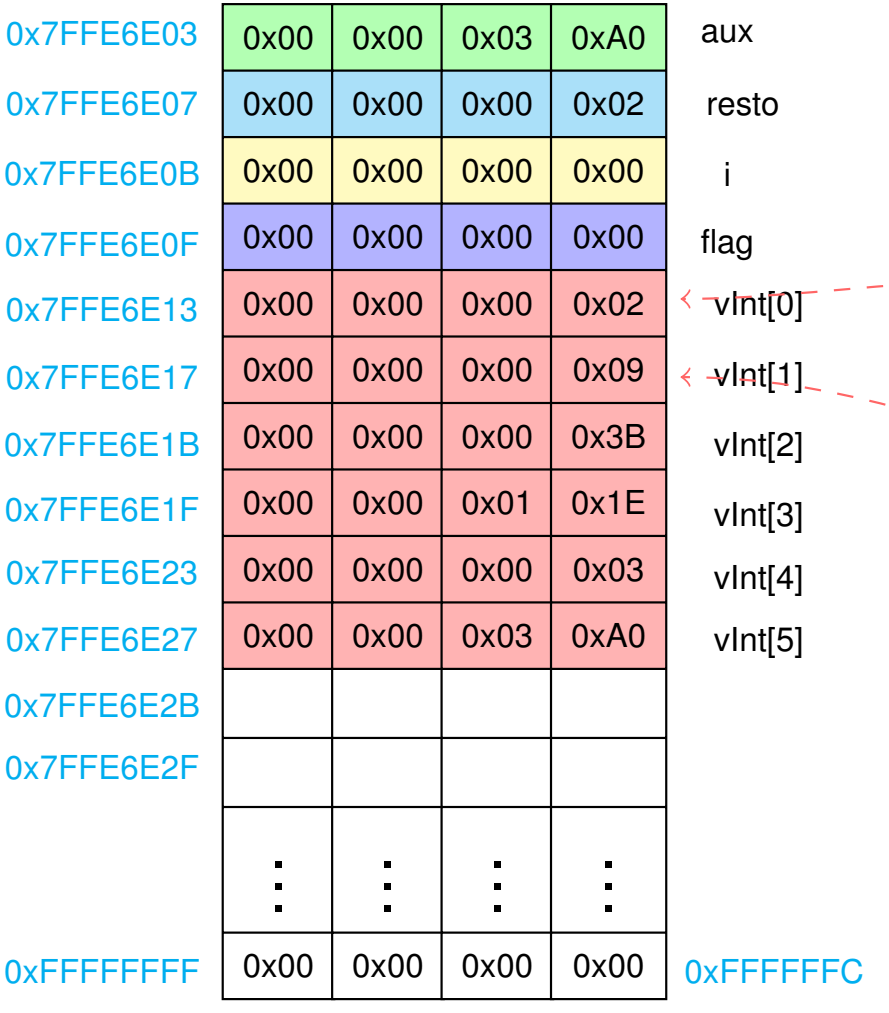
MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-2=4

MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-2=4

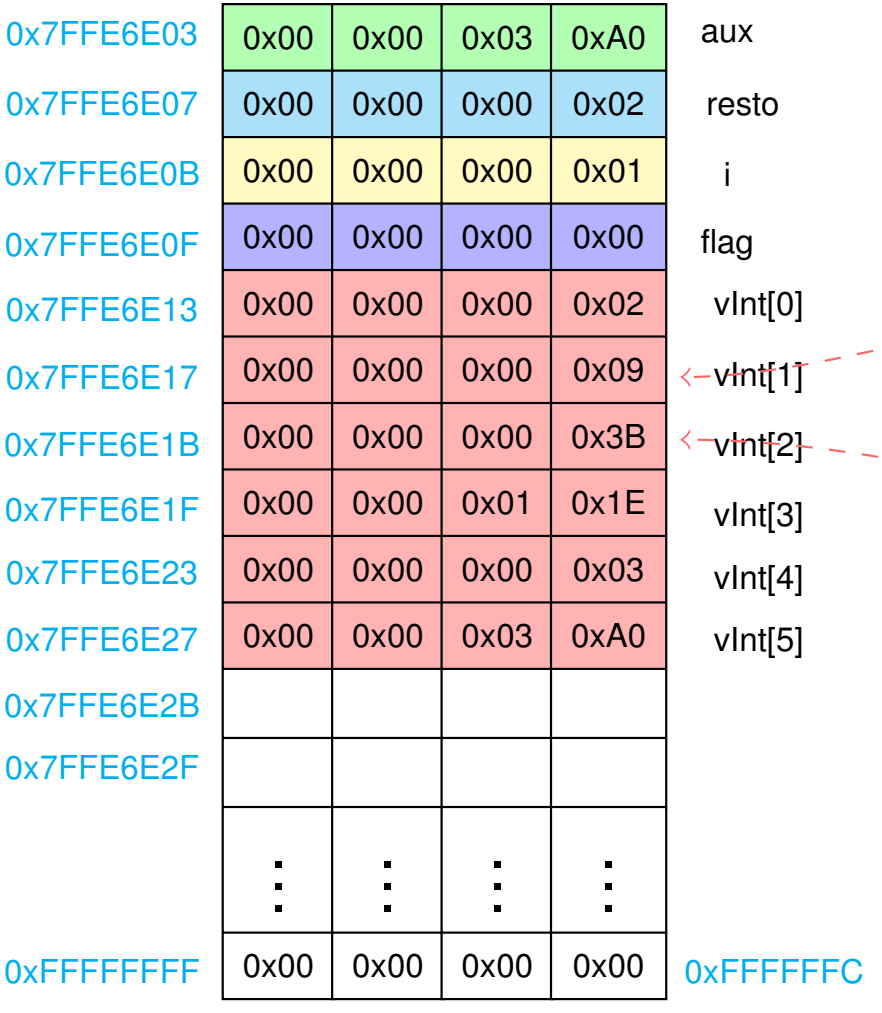
0 1

MEMORIA en ARQUITECTURA x86 32-bits


```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-2=4

MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-2=4

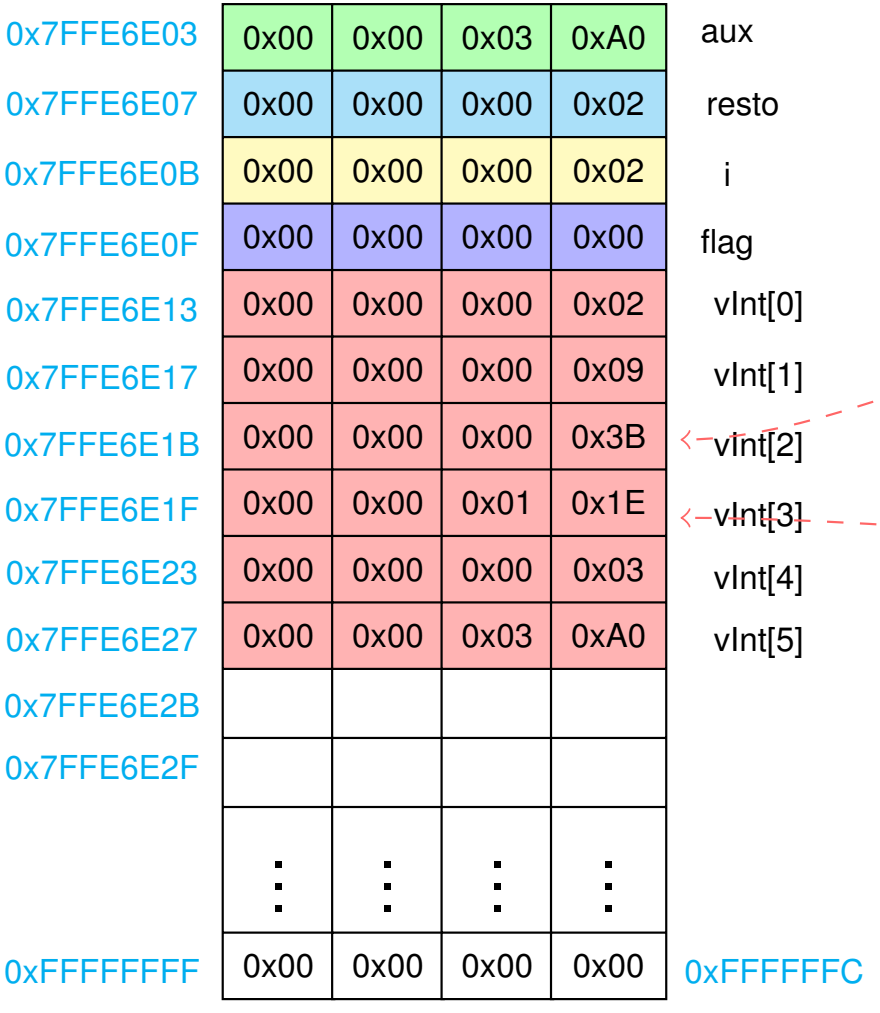
1 2

MEMORIA en ARQUITECTURA x86 32-bits


```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-2=4

MEMORIA en ARQUITECTURA x86 32-bits



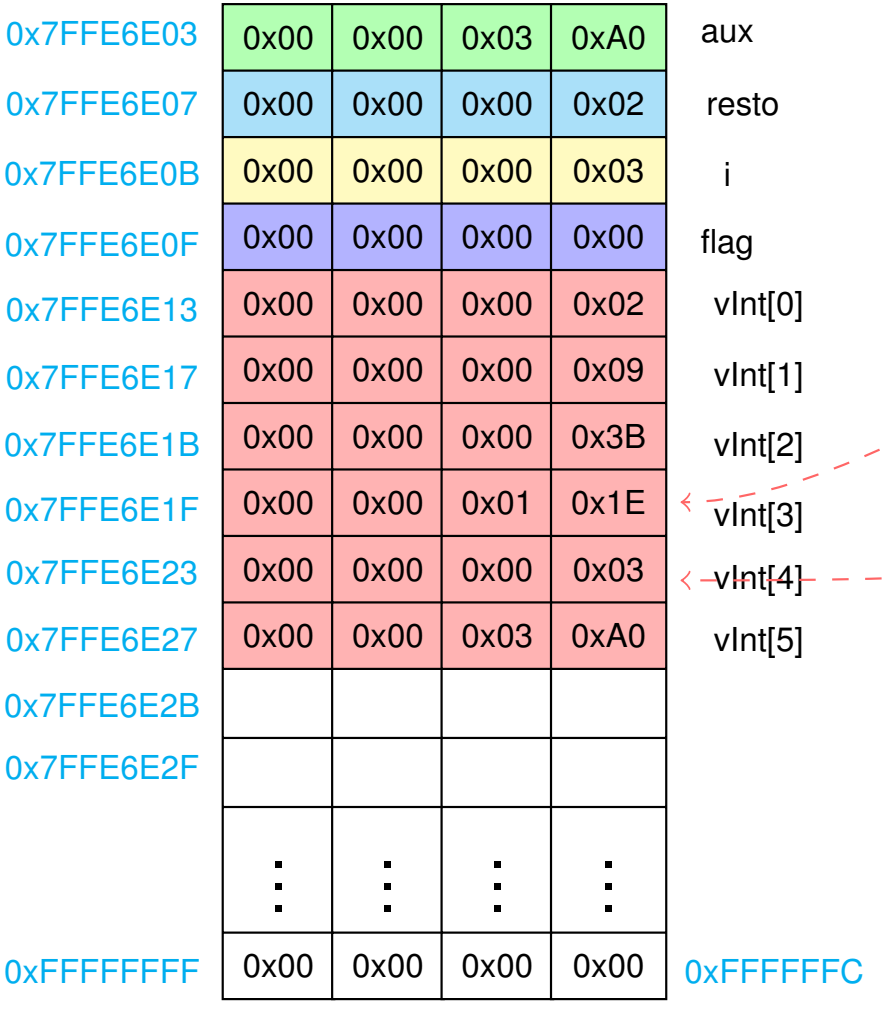

```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;                    6-2=4
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {          2                3
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

MEMORIA en ARQUITECTURA x86 32-bits


```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-2=4

MEMORIA en ARQUITECTURA x86 32-bits



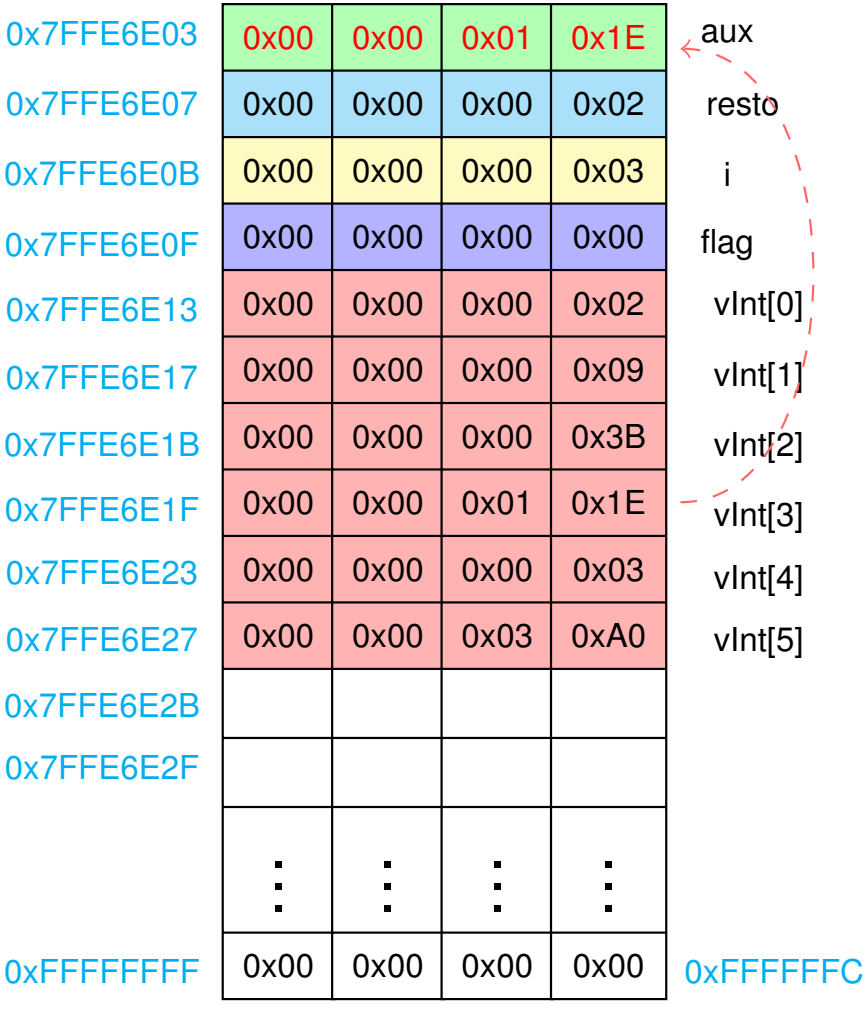

```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-2=4

3

4

MEMORIA en ARQUITECTURA x86 32-bits

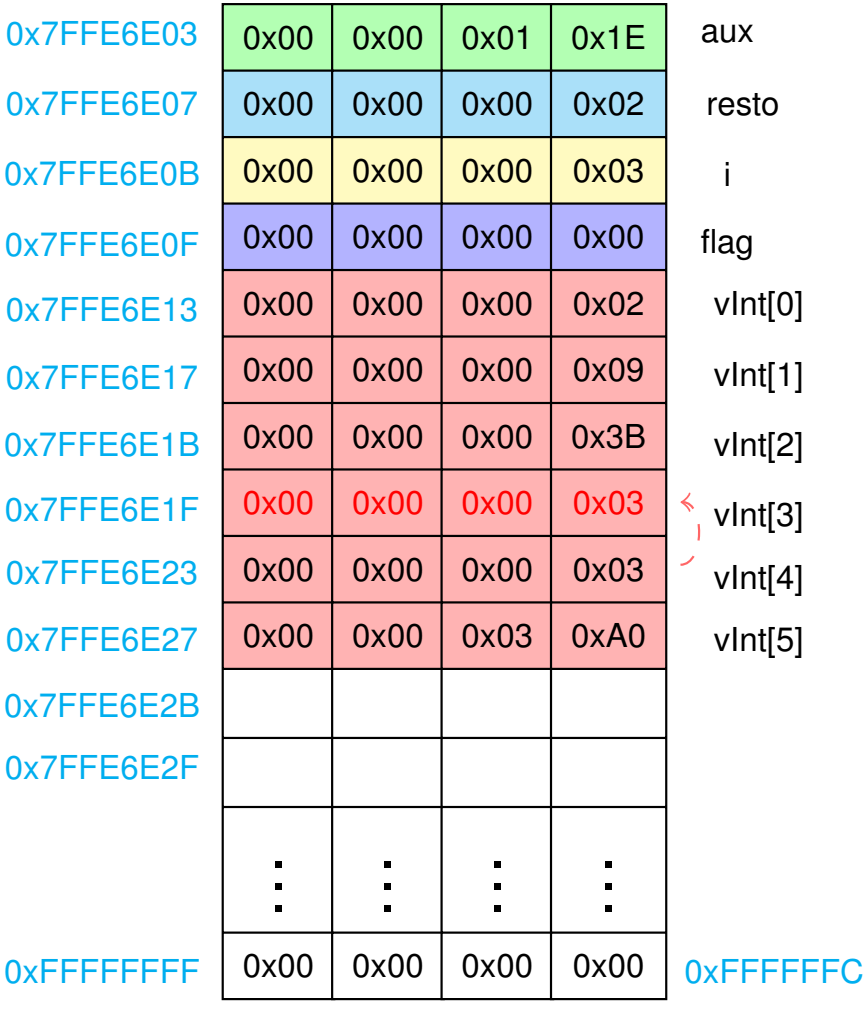



```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-2=4

3

MEMORIA en ARQUITECTURA x86 32-bits

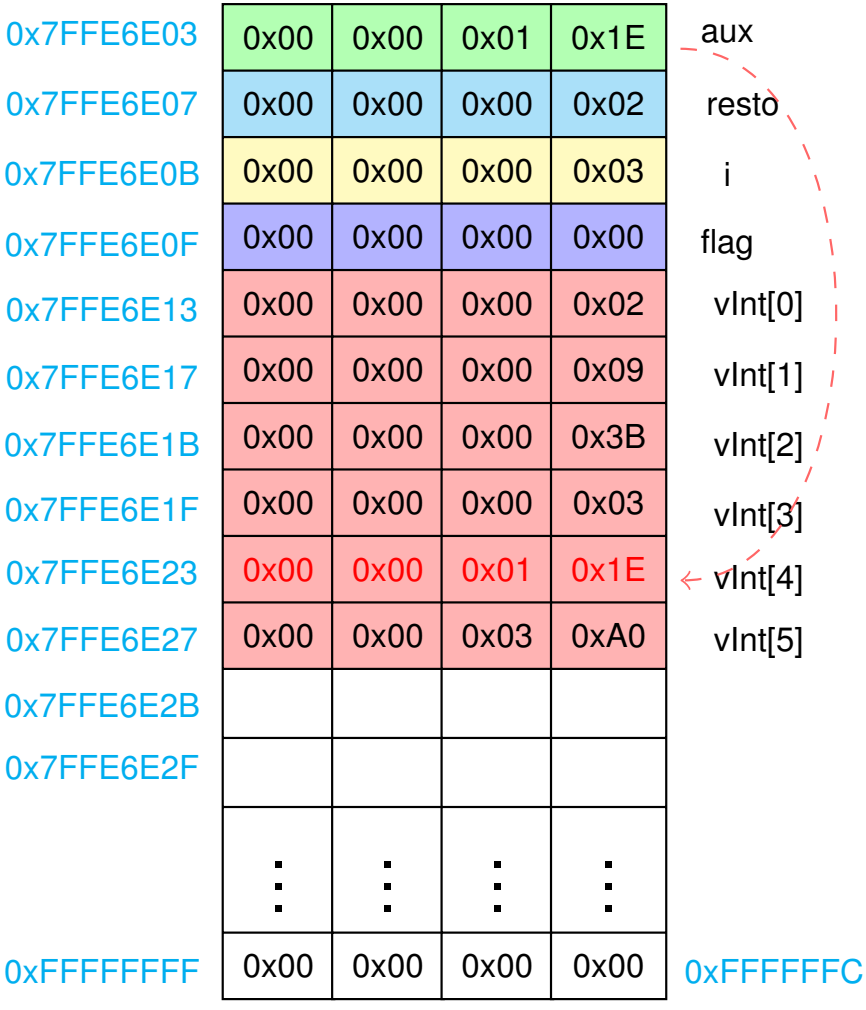



```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-2=4

vInt[3] = vInt[4]

MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-2=4

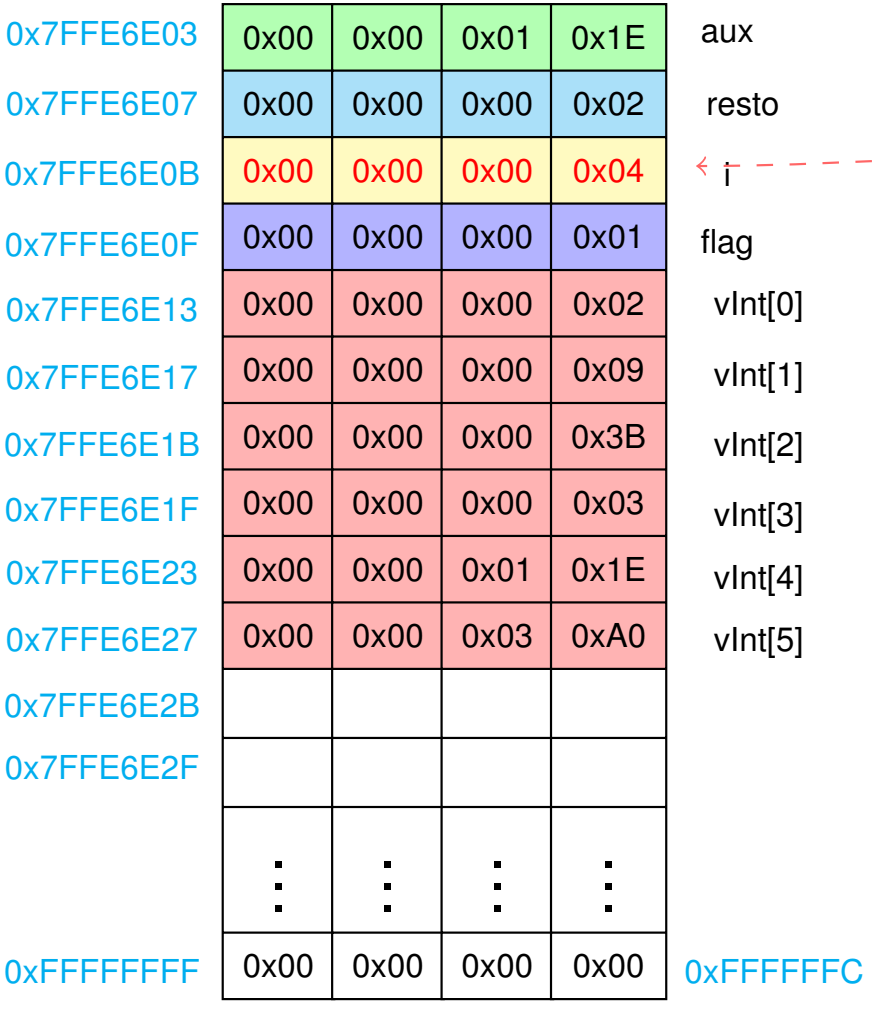
vInt[4] = aux

MEMORIA en ARQUITECTURA x86 32-bits


```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-2=4

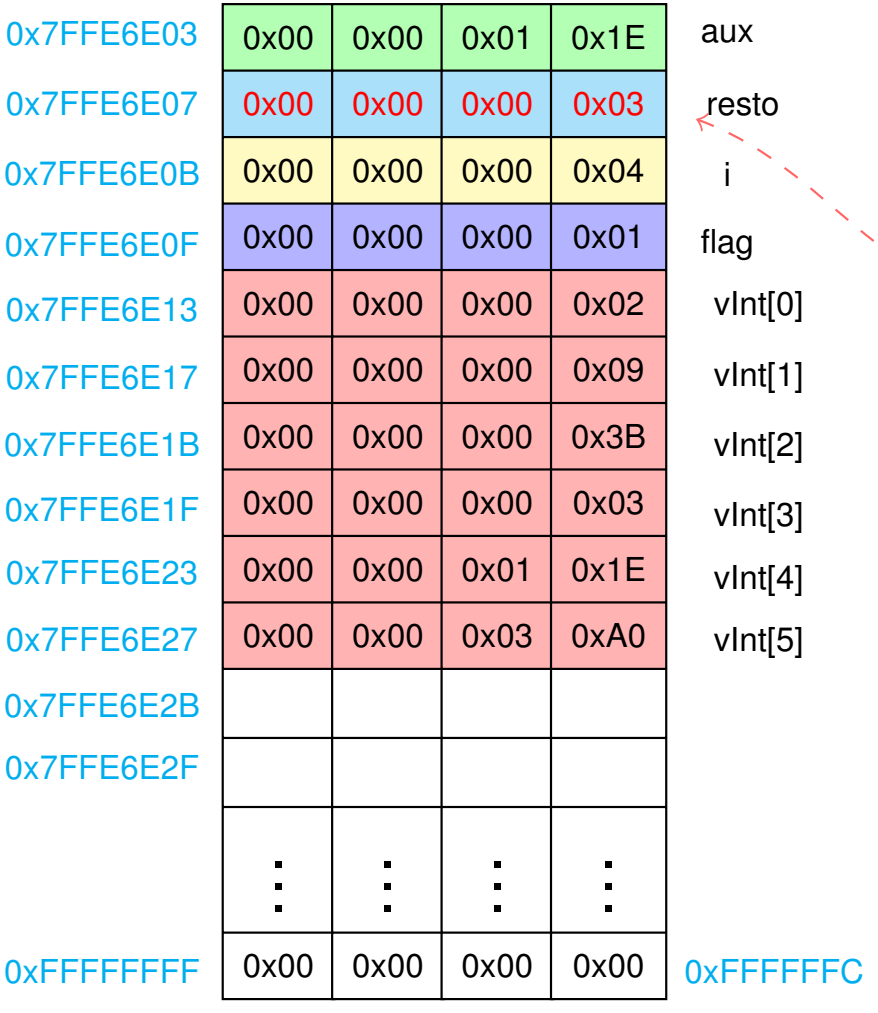
MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-2=4

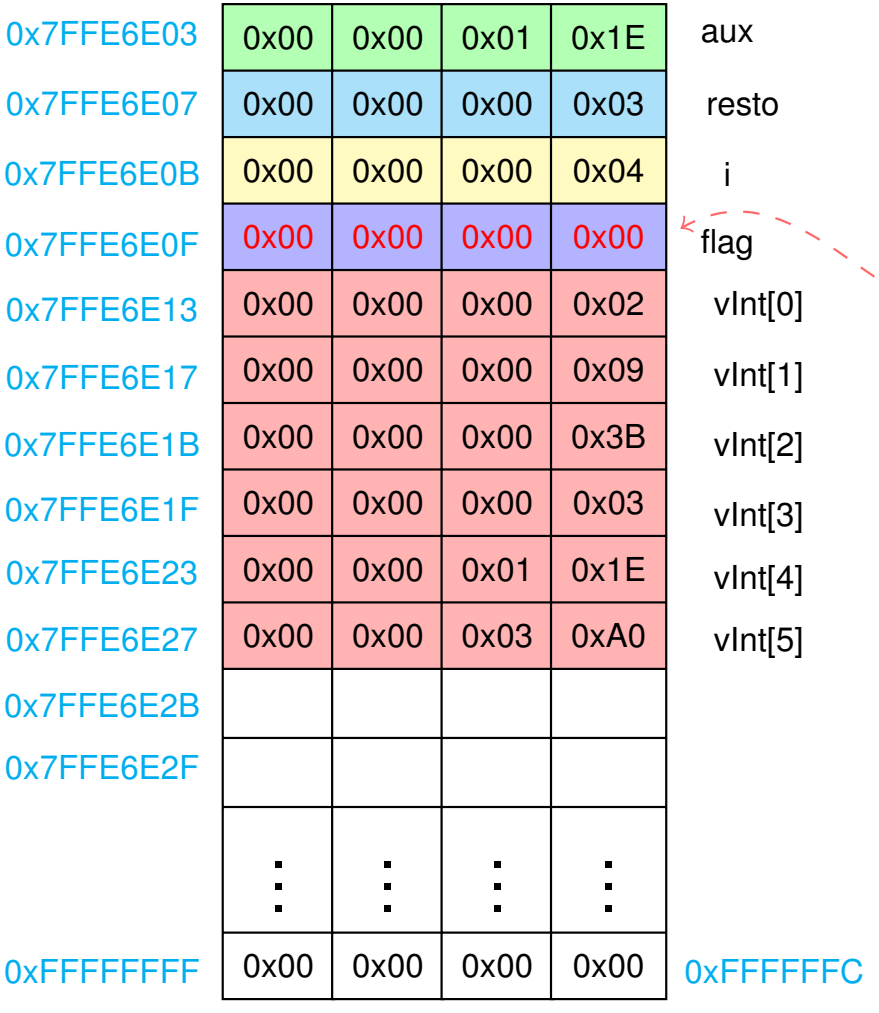
MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-2=4

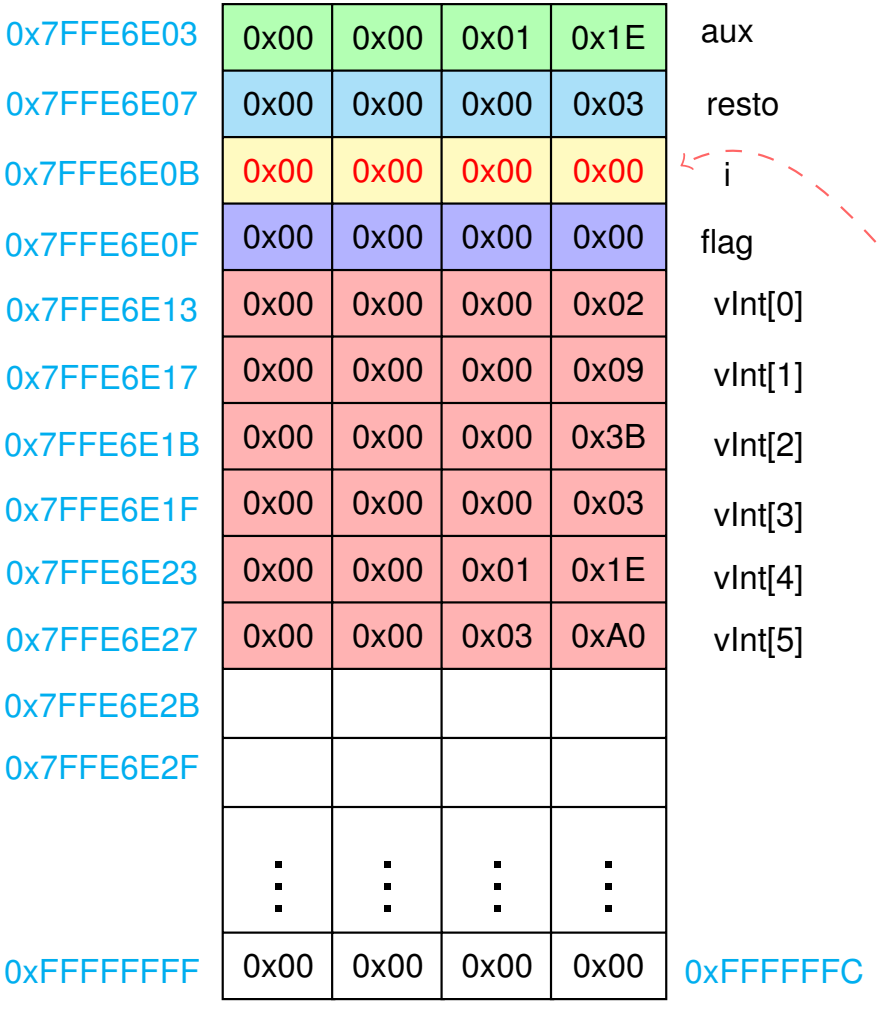
MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-2=4

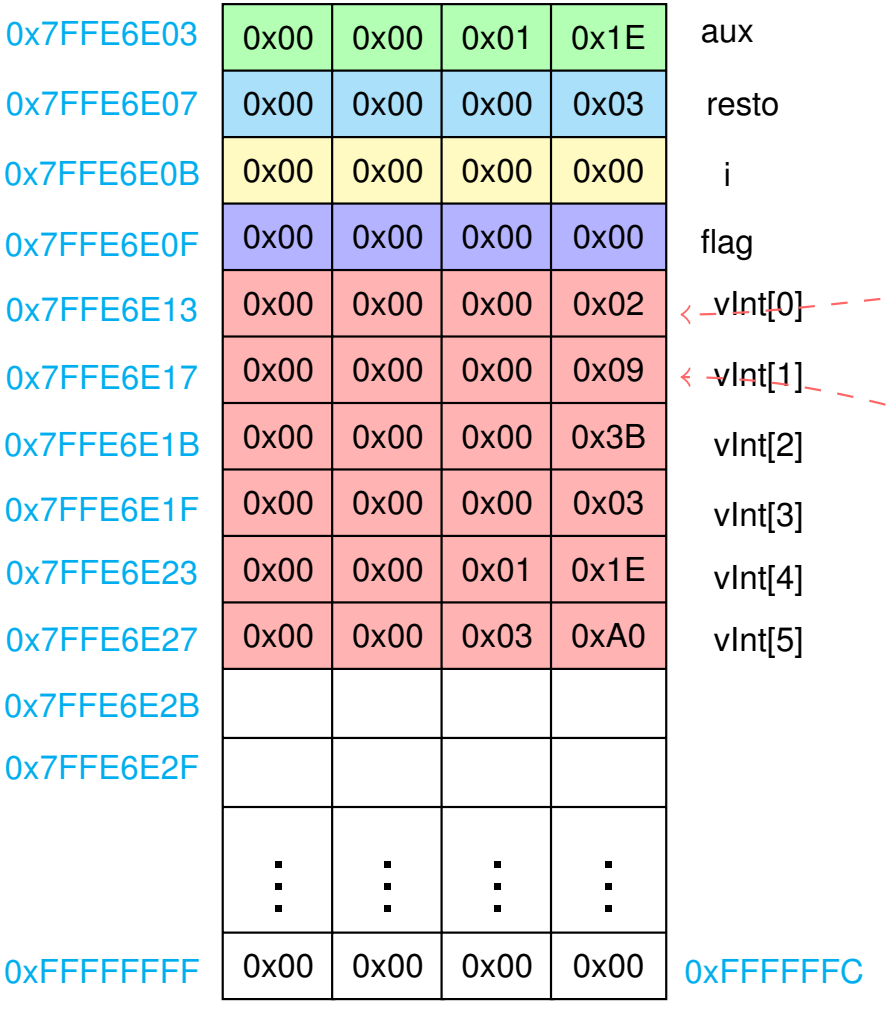
MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-3=3

MEMORIA en ARQUITECTURA x86 32-bits

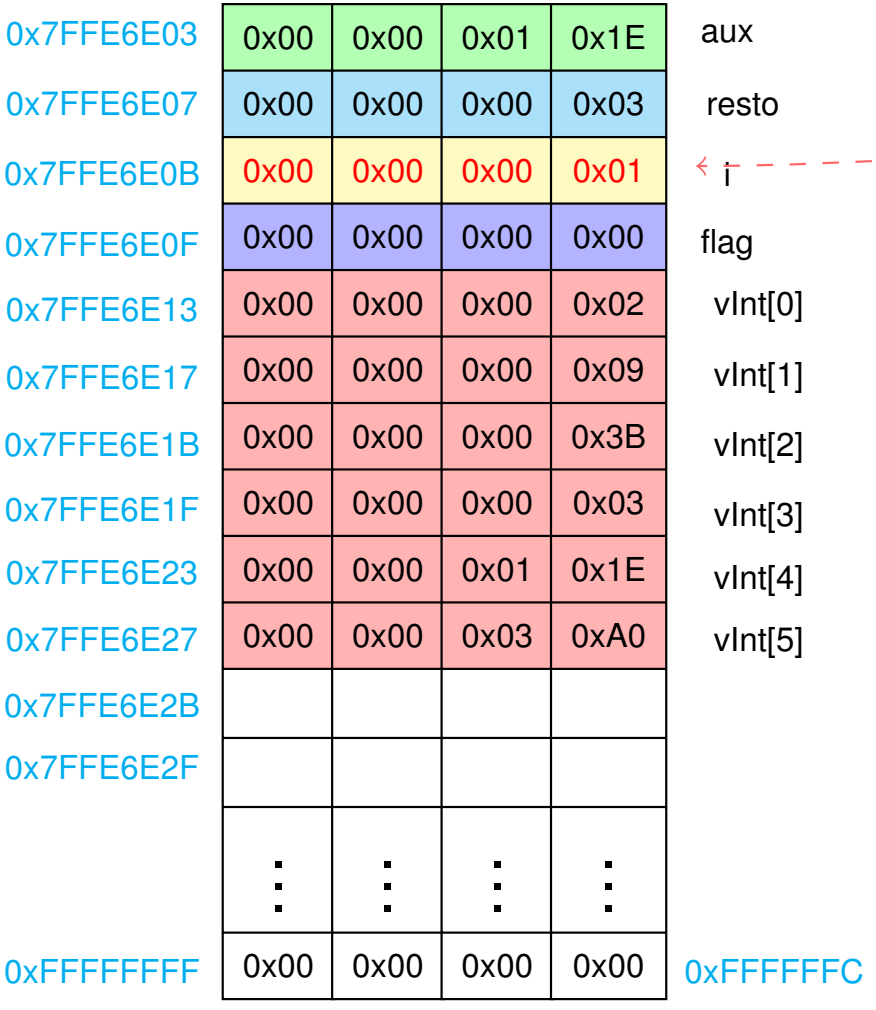



```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-3=3

0 1

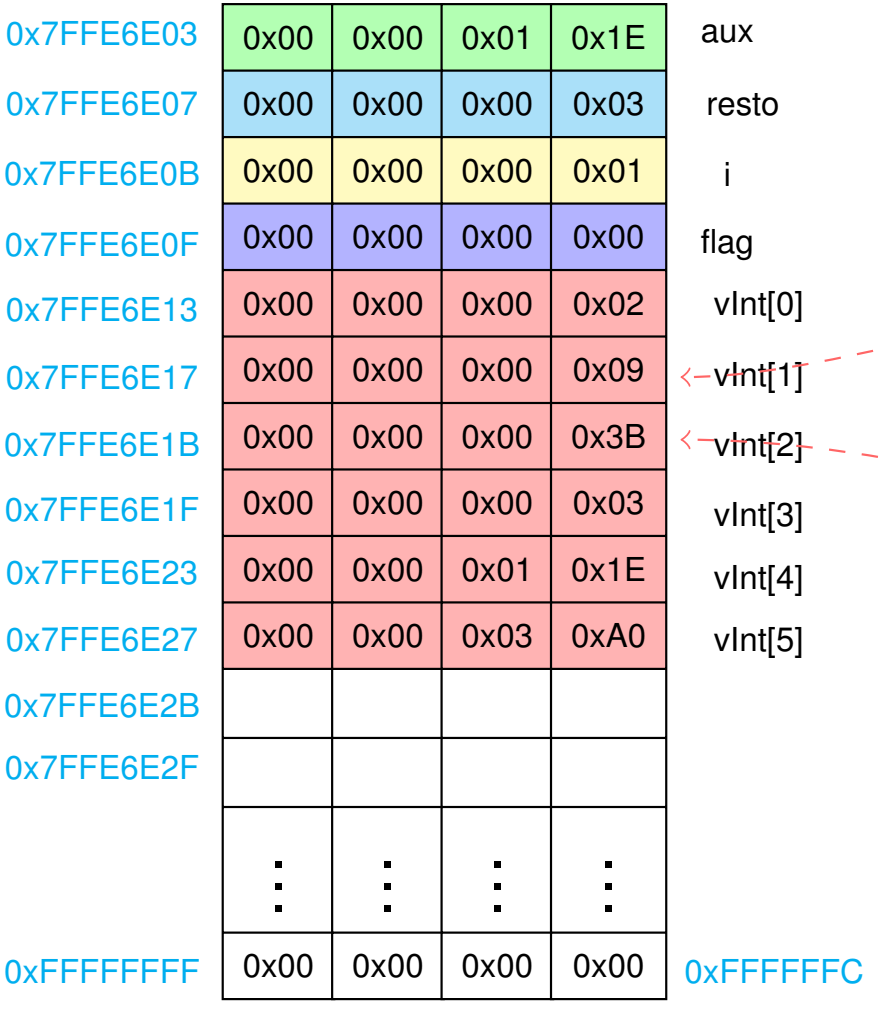
MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

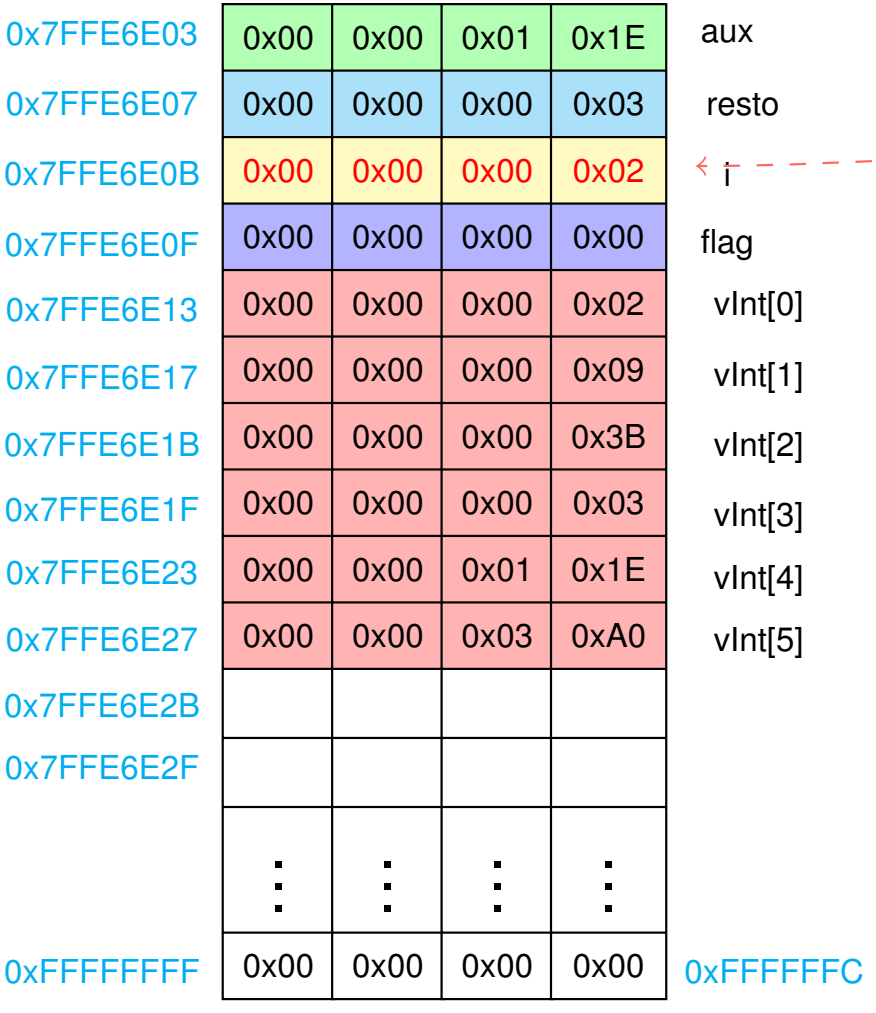
6-3=3

MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;                    6-3=3
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {          1              2
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

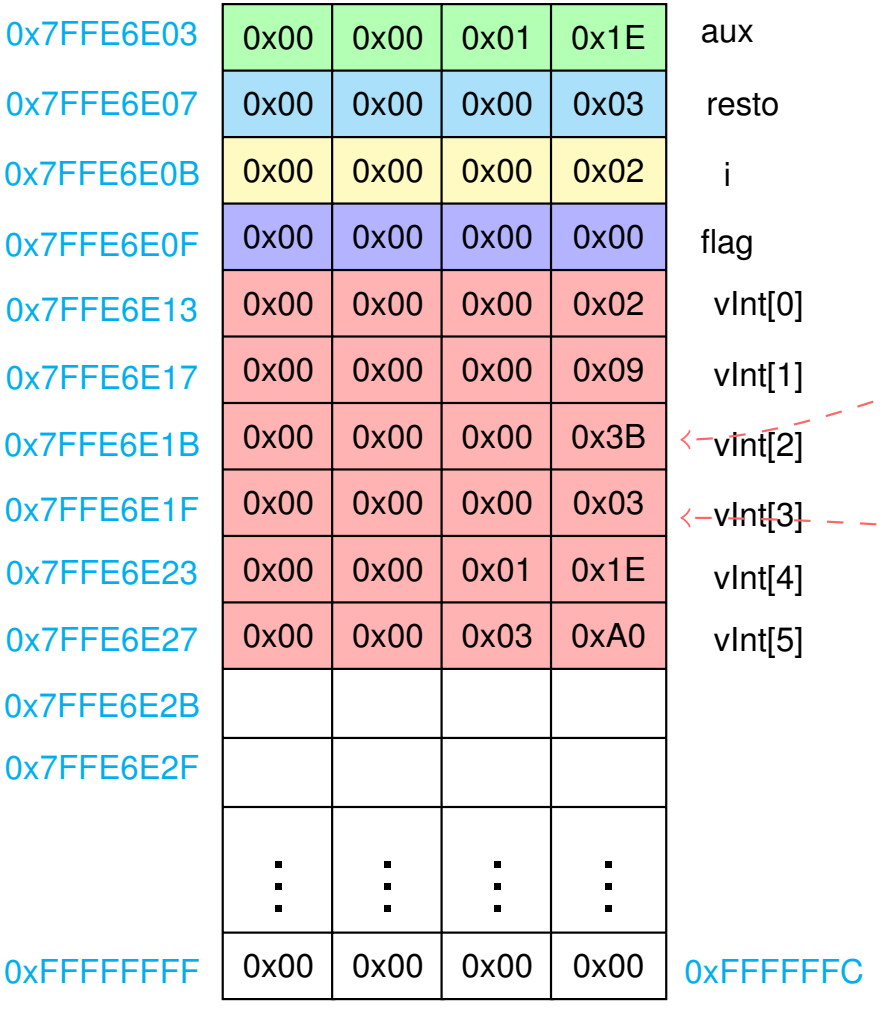
MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

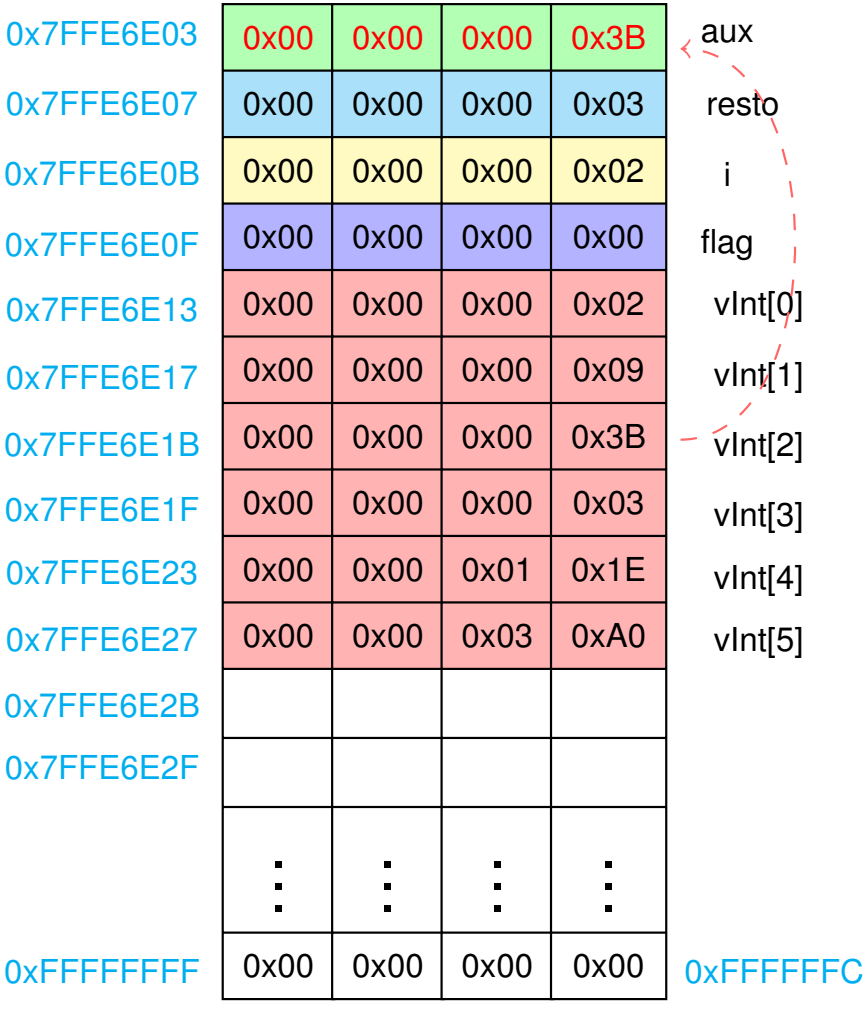
6-3=3

MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;                    6-3=3
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {          2                3
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;                6-3=3
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {                2
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

MEMORIA en ARQUITECTURA x86 32-bits

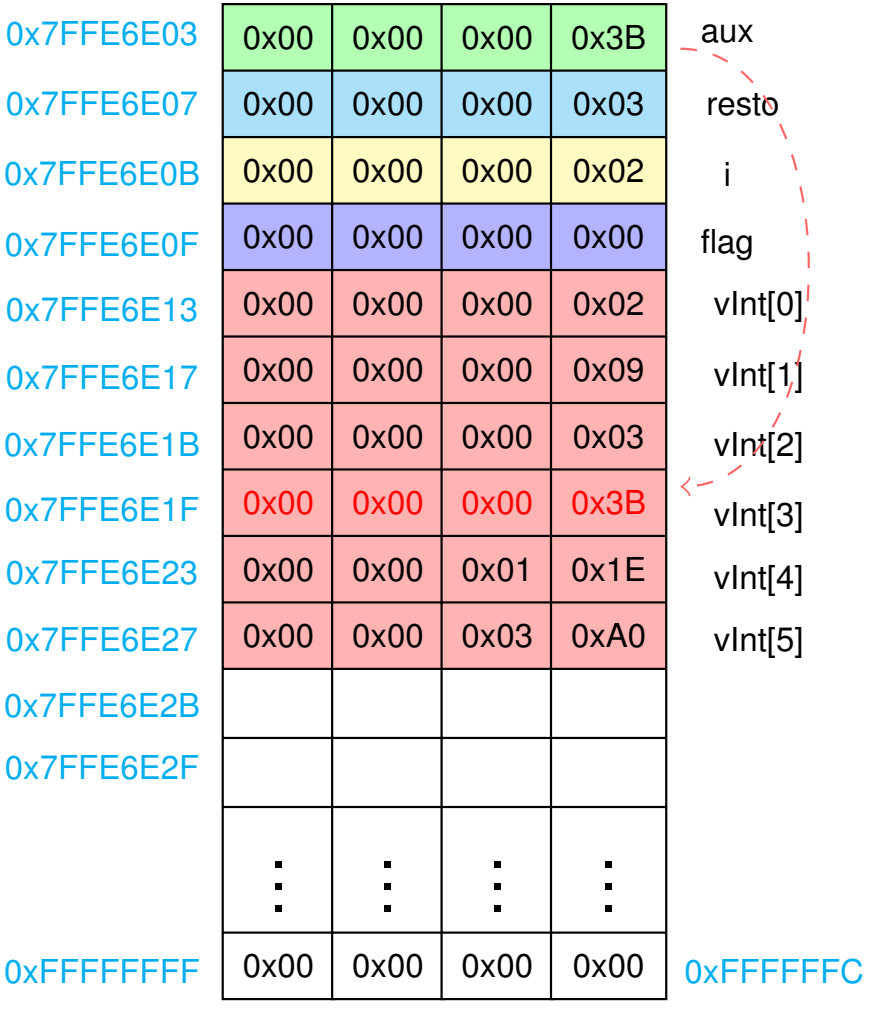
0x7FFE6E03	0x00	0x00	0x00	0x3B	aux
0x7FFE6E07	0x00	0x00	0x00	0x03	resto
0x7FFE6E0B	0x00	0x00	0x00	0x02	i
0x7FFE6E0F	0x00	0x00	0x00	0x00	flag
0x7FFE6E13	0x00	0x00	0x00	0x02	vInt[0]
0x7FFE6E17	0x00	0x00	0x00	0x09	vInt[1]
0x7FFE6E1B	0x00	0x00	0x00	0x03	vInt[2]
0x7FFE6E1F	0x00	0x00	0x00	0x03	vInt[3]
0x7FFE6E23	0x00	0x00	0x01	0x1E	vInt[4]
0x7FFE6E27	0x00	0x00	0x03	0xA0	vInt[5]
0x7FFE6E2B					
0x7FFE6E2F					
	⋮	⋮	⋮	⋮	
0xFFFFFFFF	0x00	0x00	0x00	0x00	0xFFFFFFC

```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-3=3

vInt[2] = vInt[3]

MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-3=3

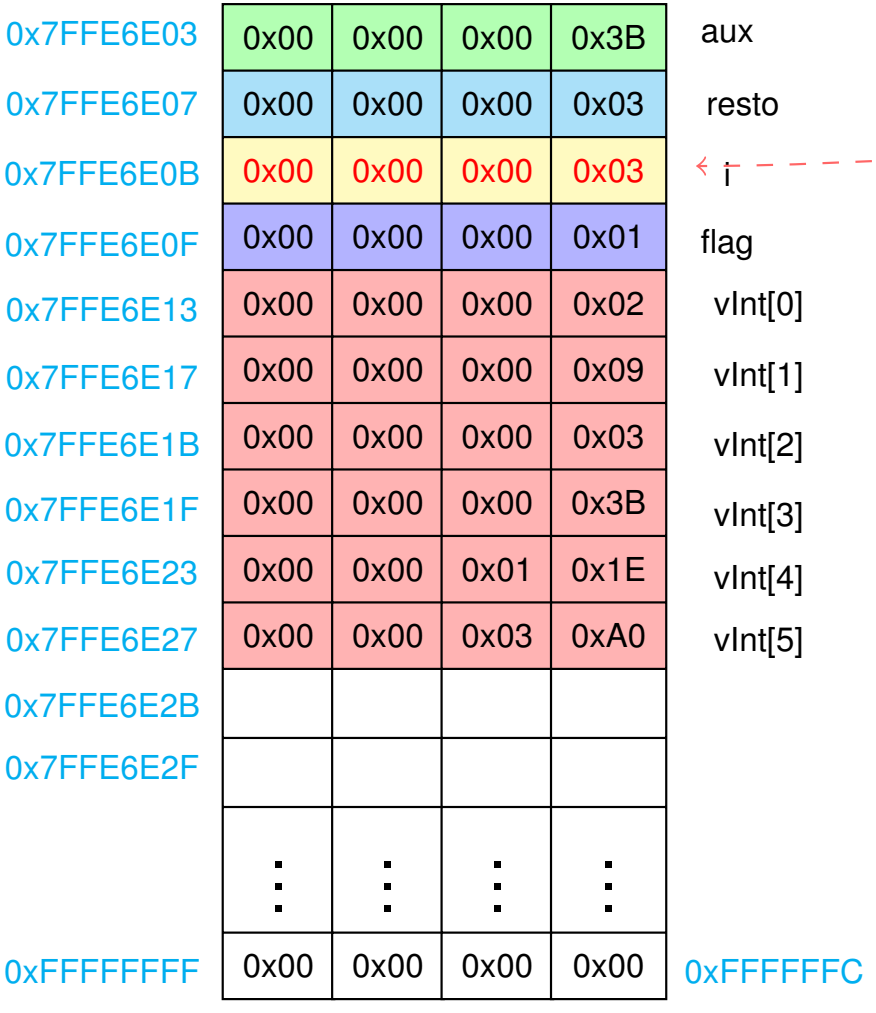
vInt[3] = aux

MEMORIA en ARQUITECTURA x86 32-bits


```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-3=3

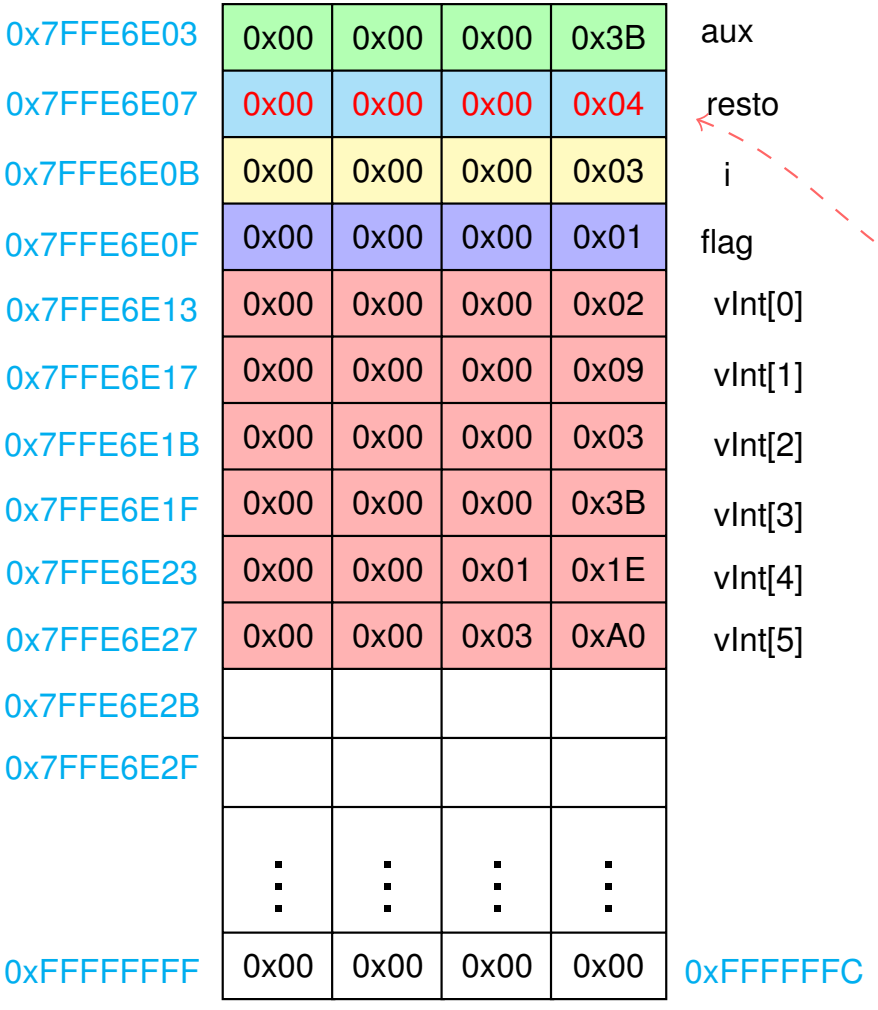
MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-3=3

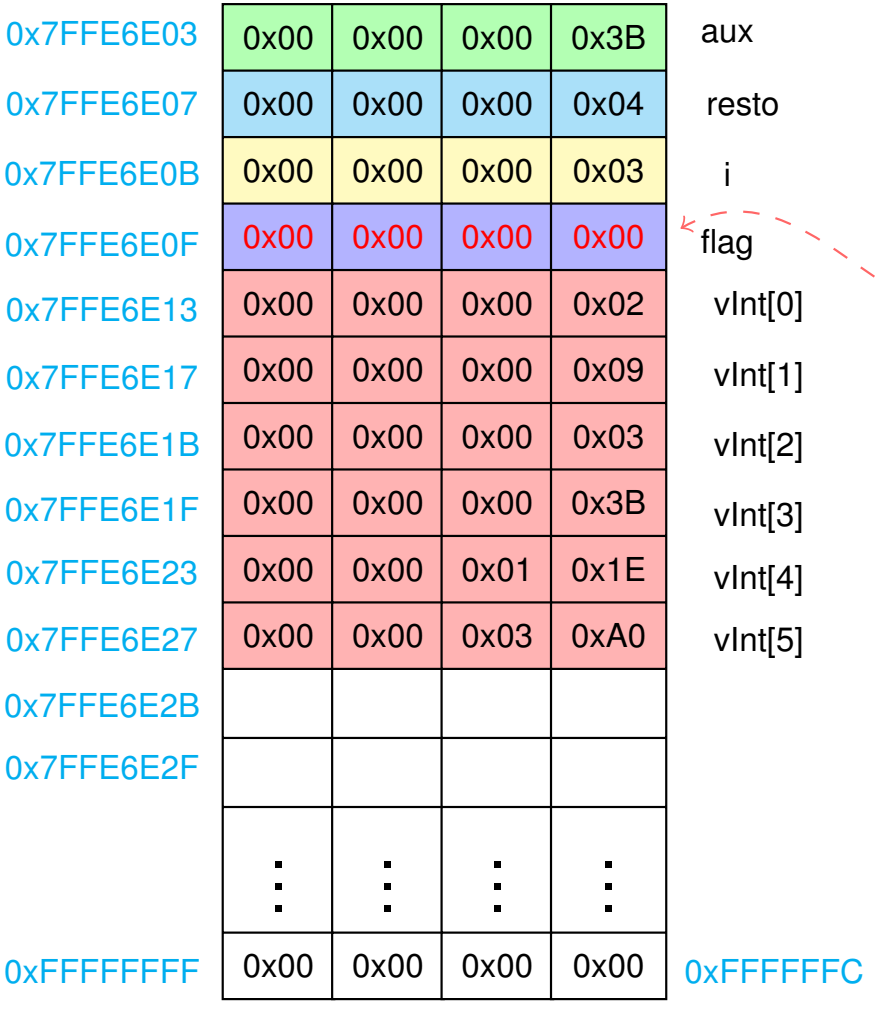
MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-3=3

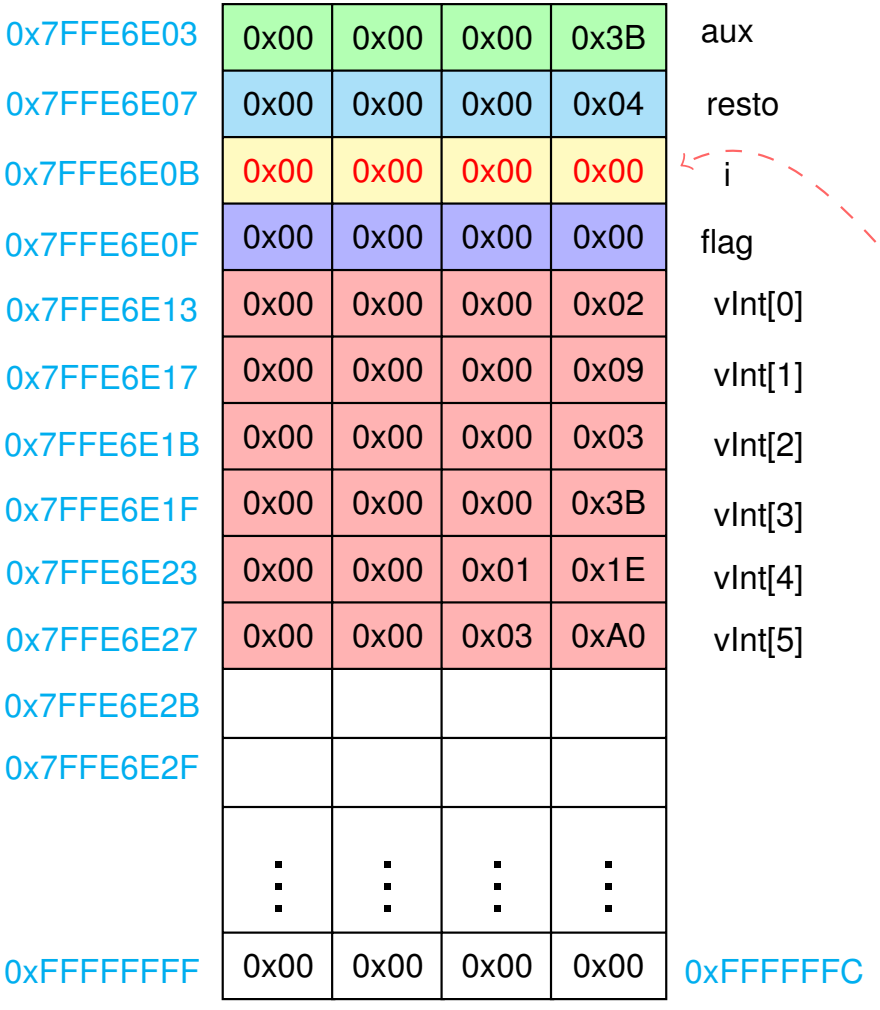
MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-3=3

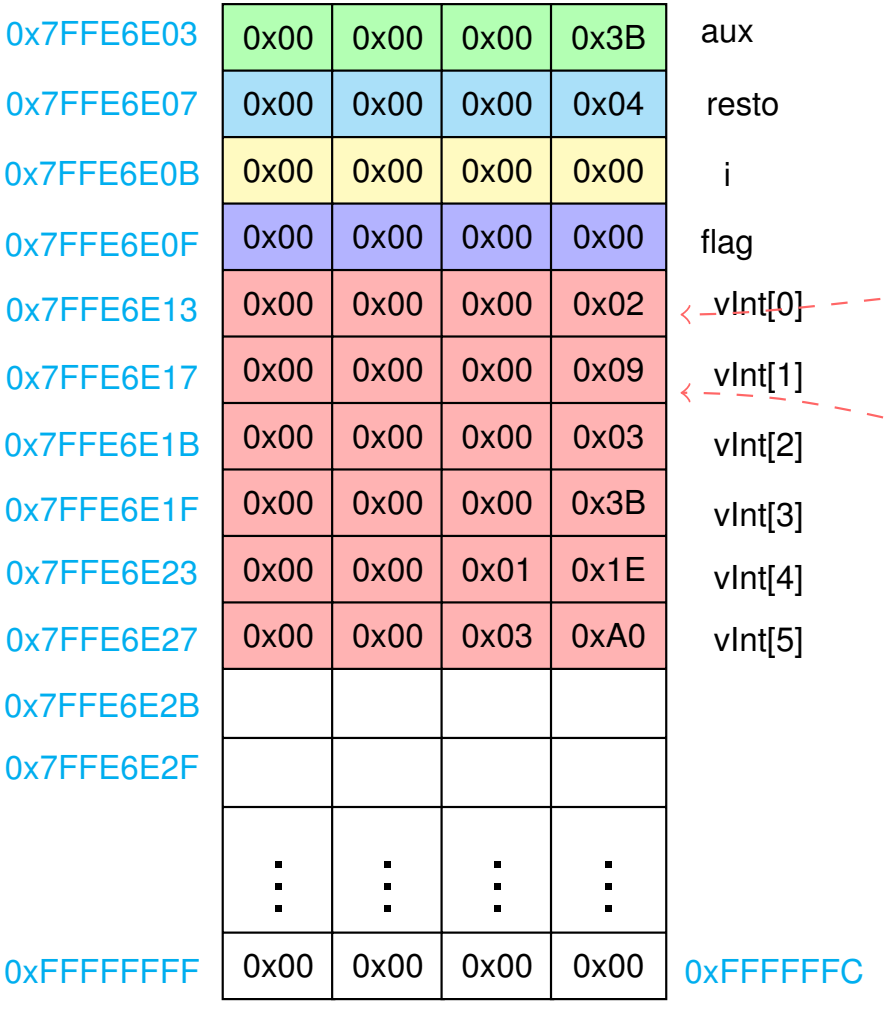
MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-4=2

MEMORIA en ARQUITECTURA x86 32-bits

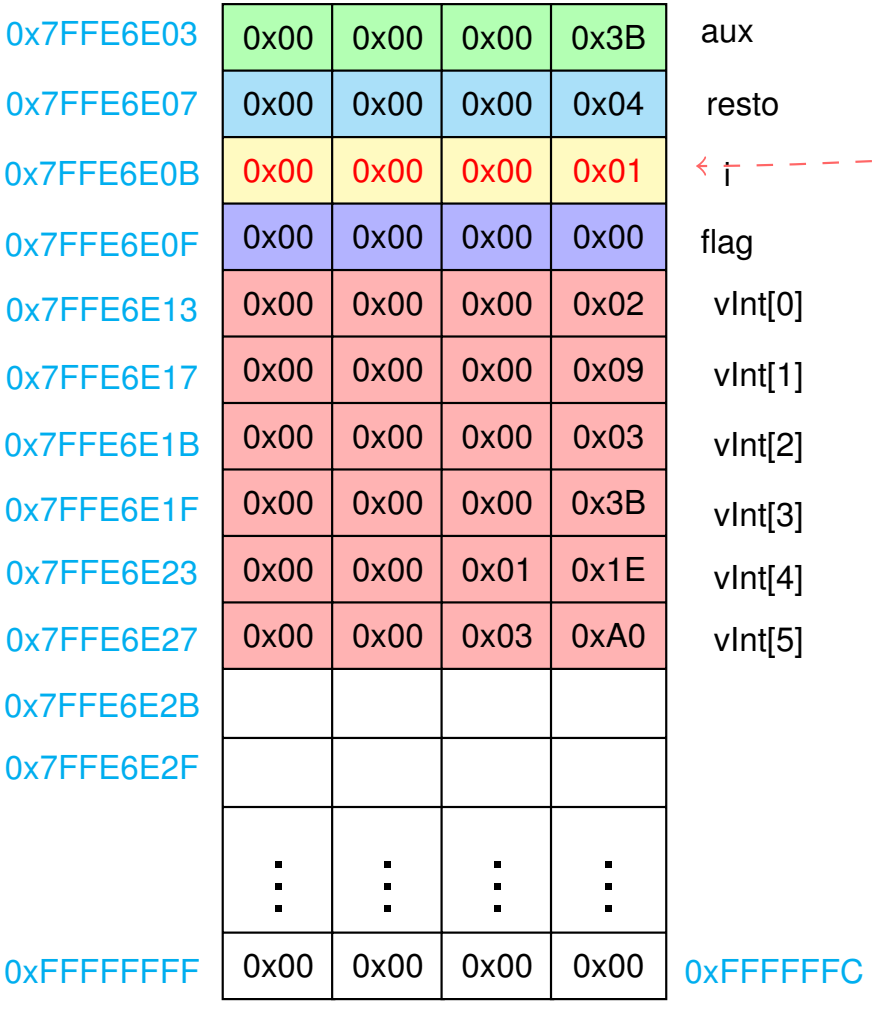



```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-4=2

0 1

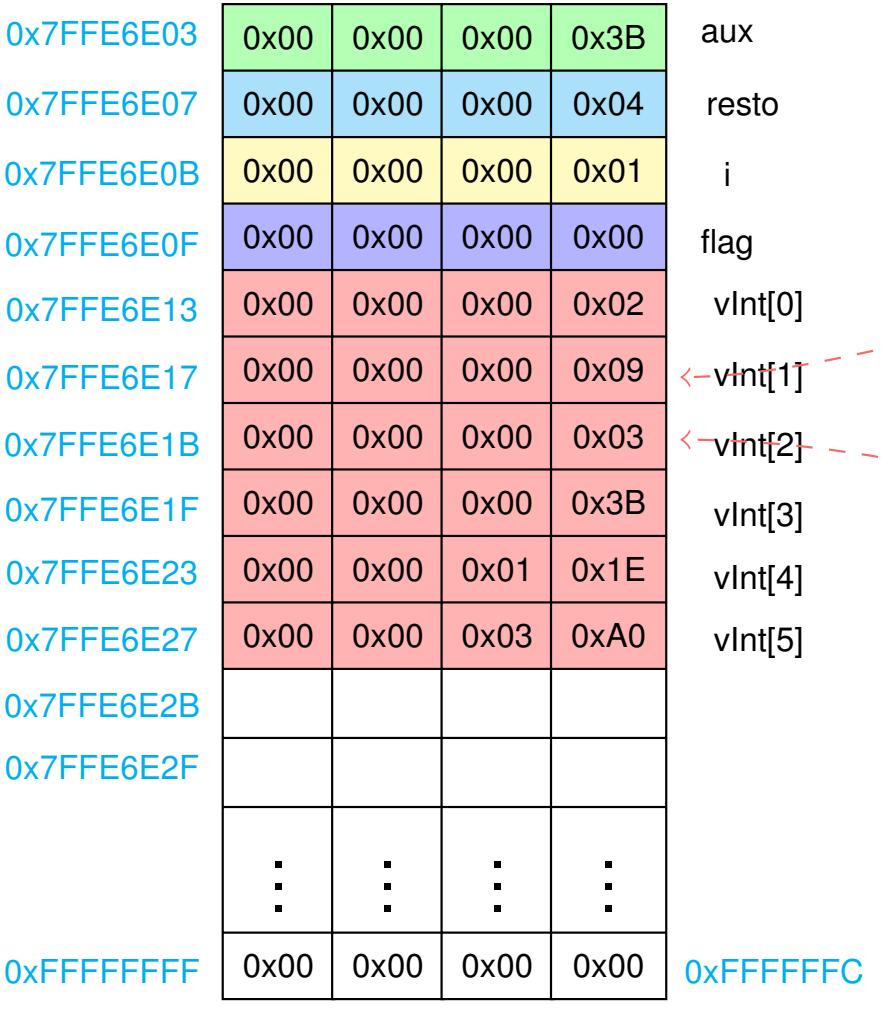
MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

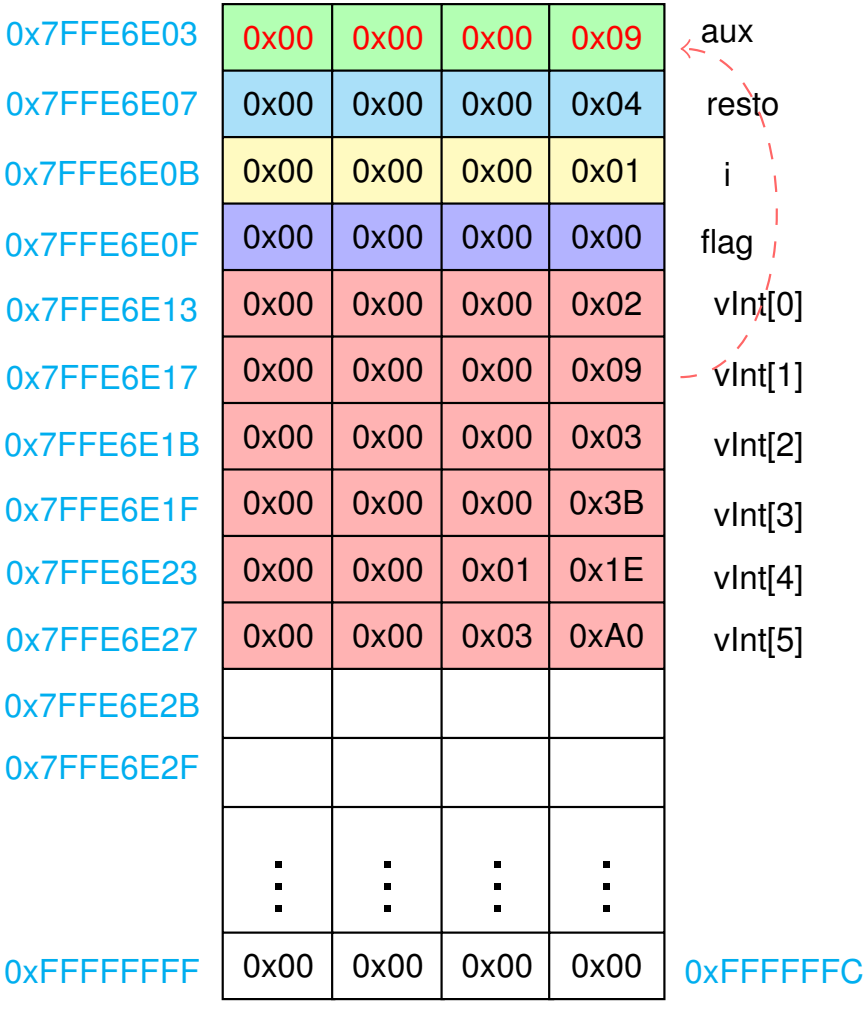
6-4=2

MEMORIA en ARQUITECTURA x86 32-bits



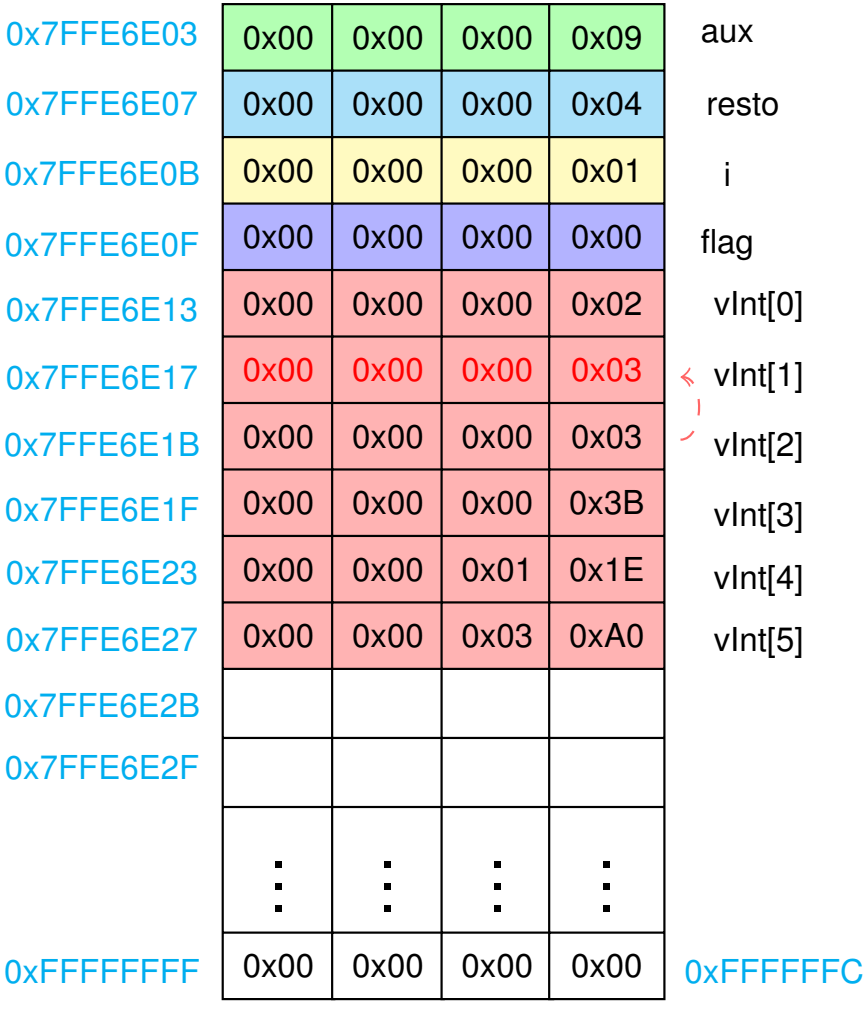

```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;                    6-4=2
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {          1              2
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;                    6-4=2
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {                  1
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

MEMORIA en ARQUITECTURA x86 32-bits

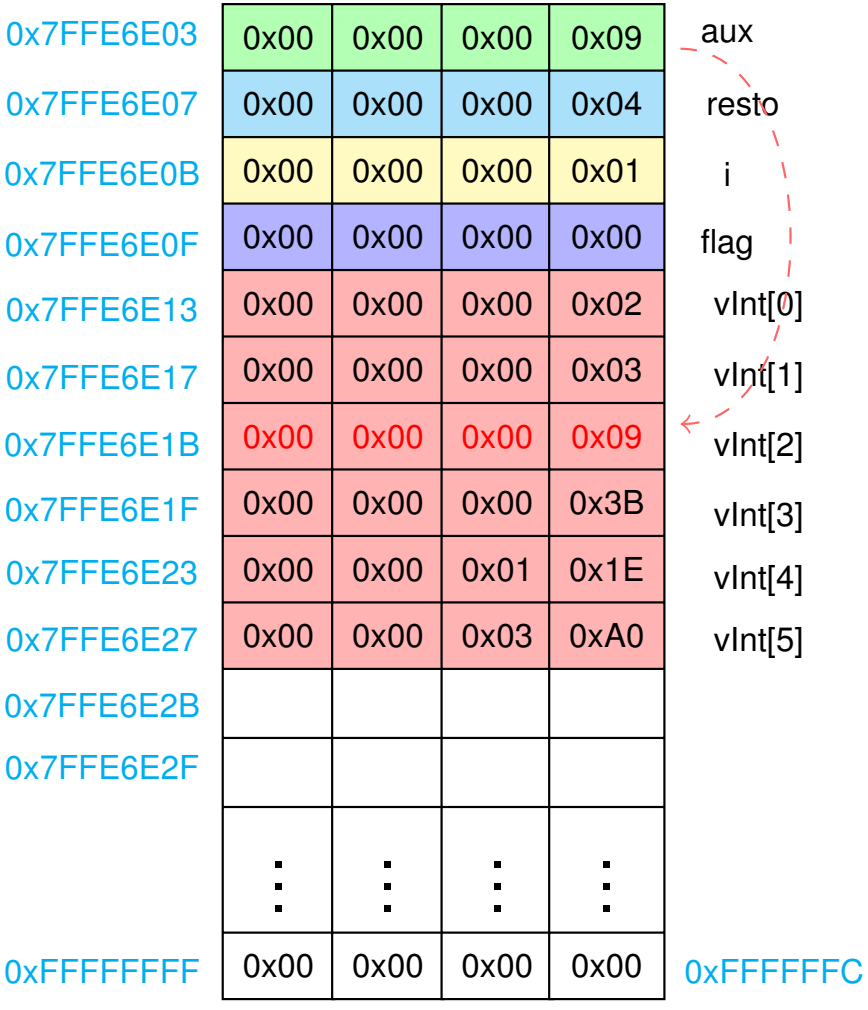



```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-4=2

vInt[1] = vInt[2]

MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

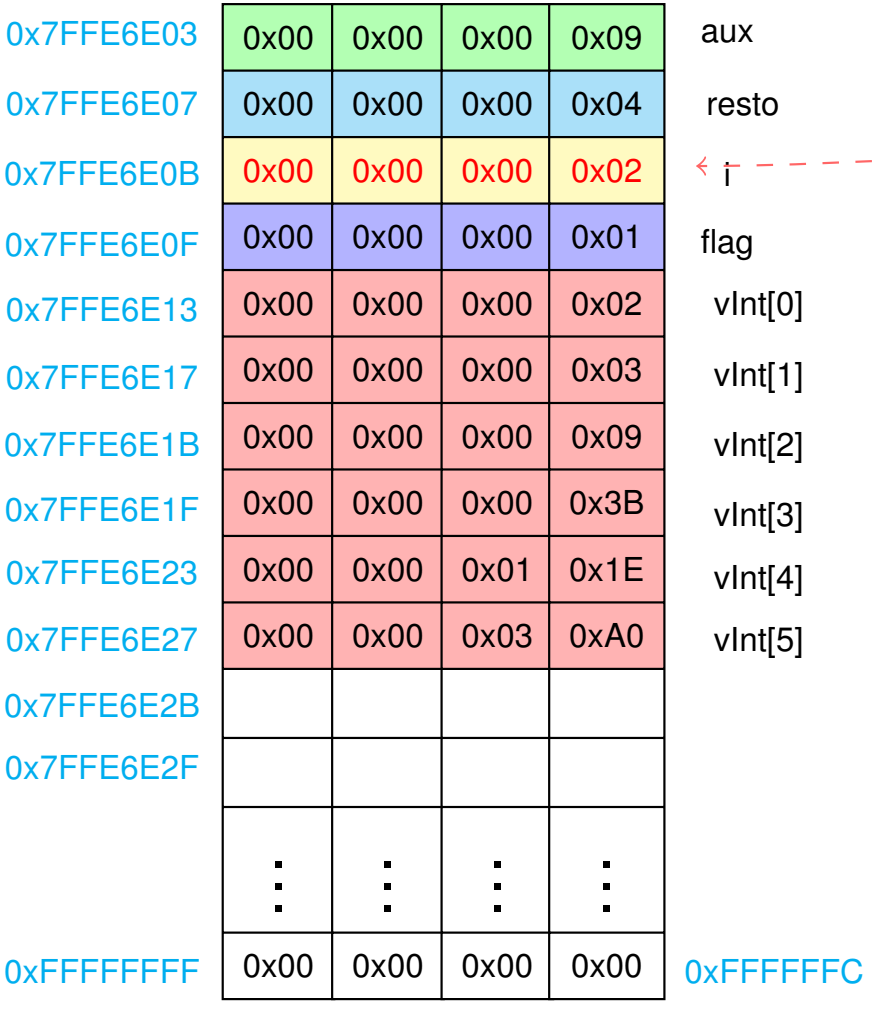
6-4=2

vInt[2] = aux

MEMORIA en ARQUITECTURA x86 32-bits


```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;                    6-4=2
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

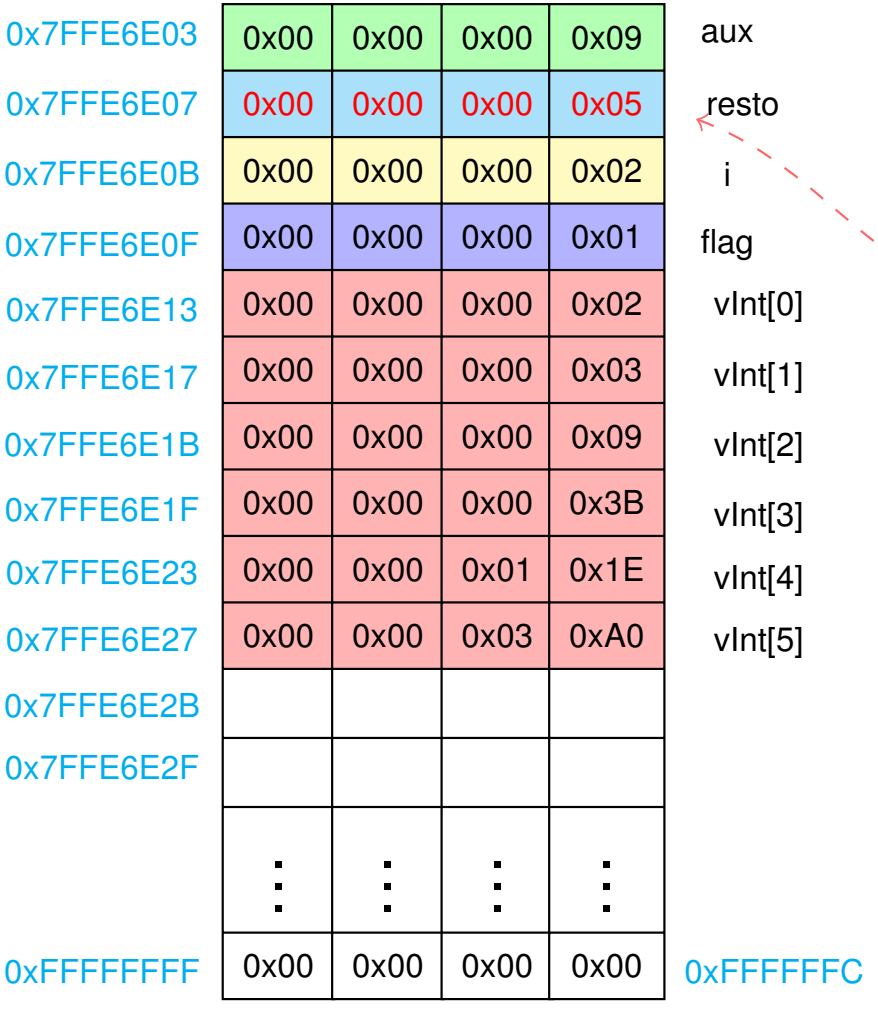
MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-4=2

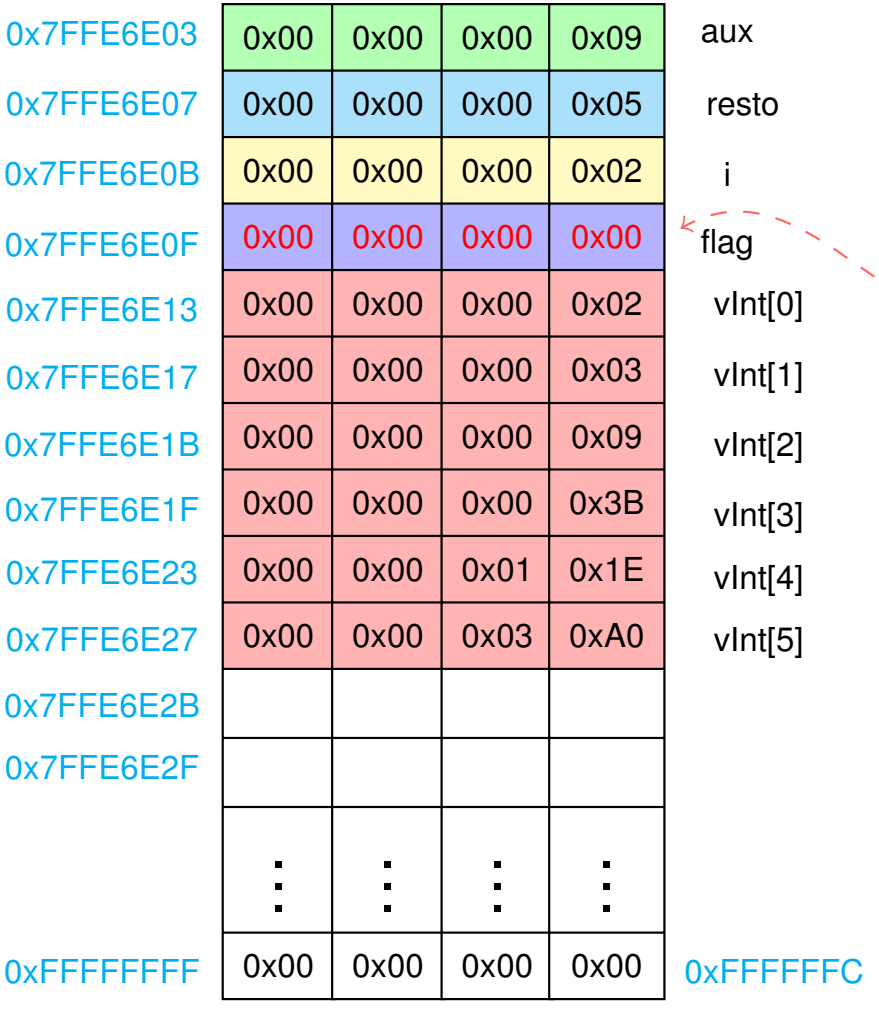
MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-4=2

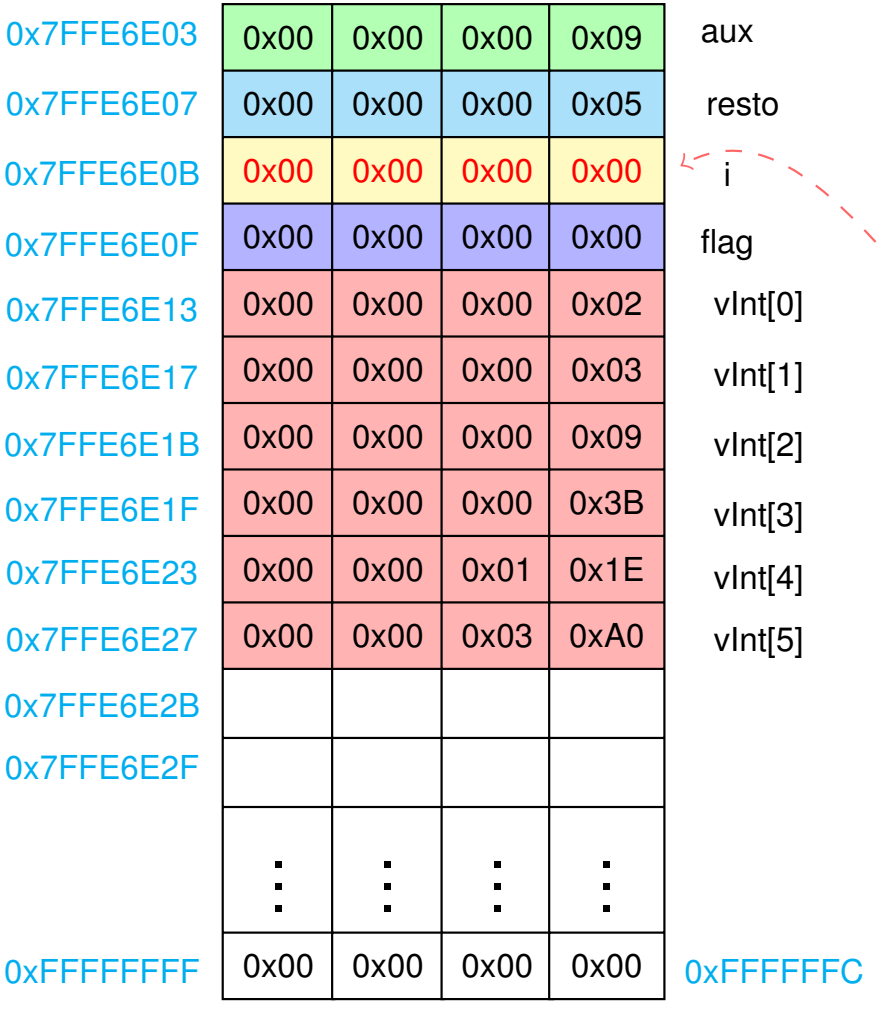
MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-4=2

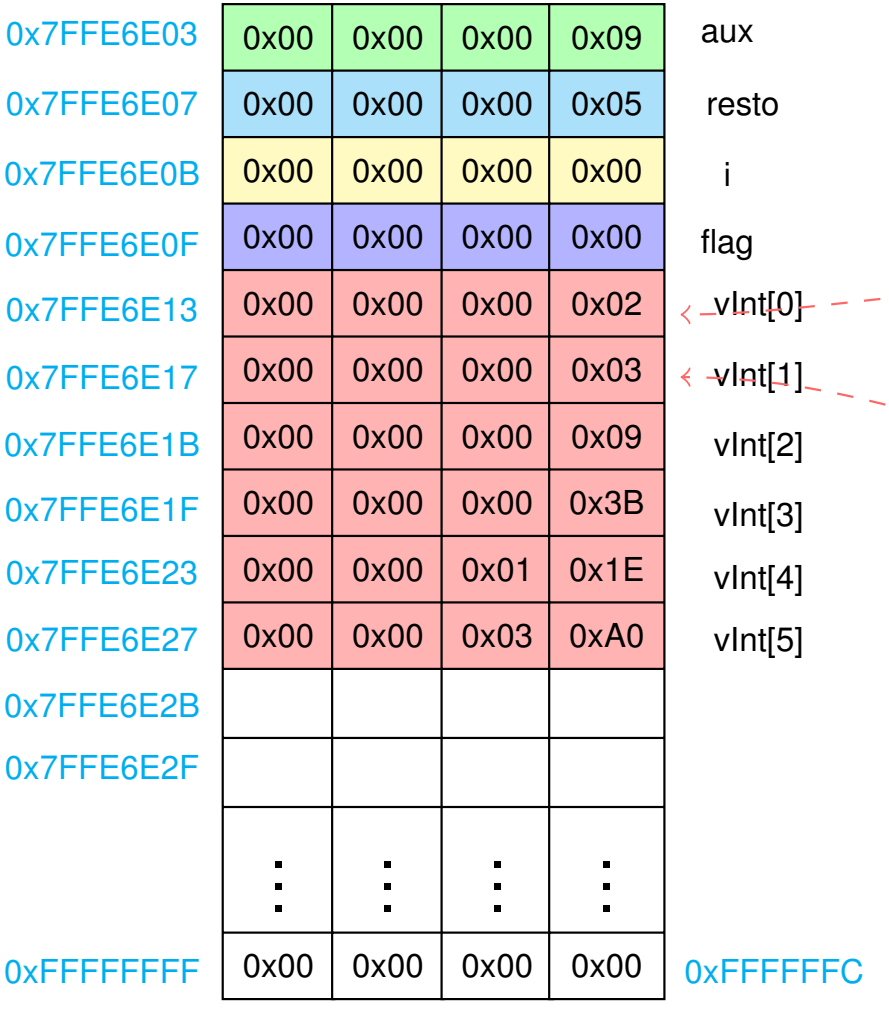
MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-5=1

MEMORIA en ARQUITECTURA x86 32-bits

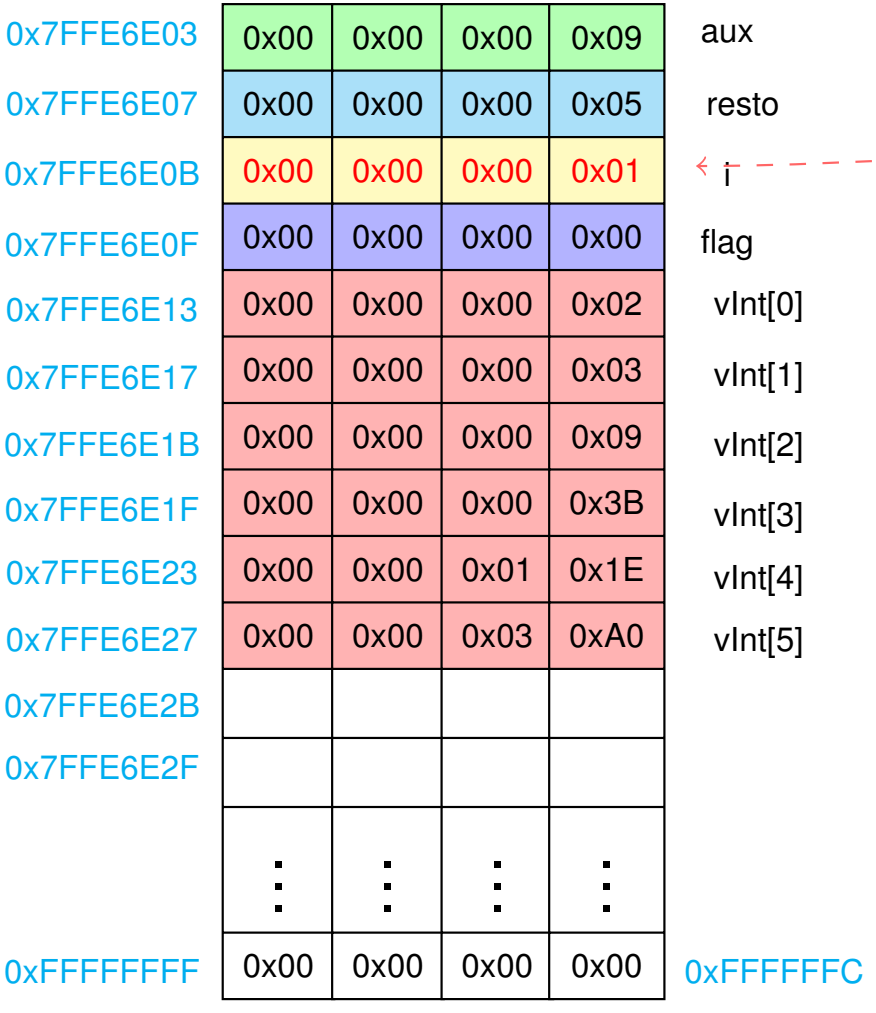



```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-5=1

0 1

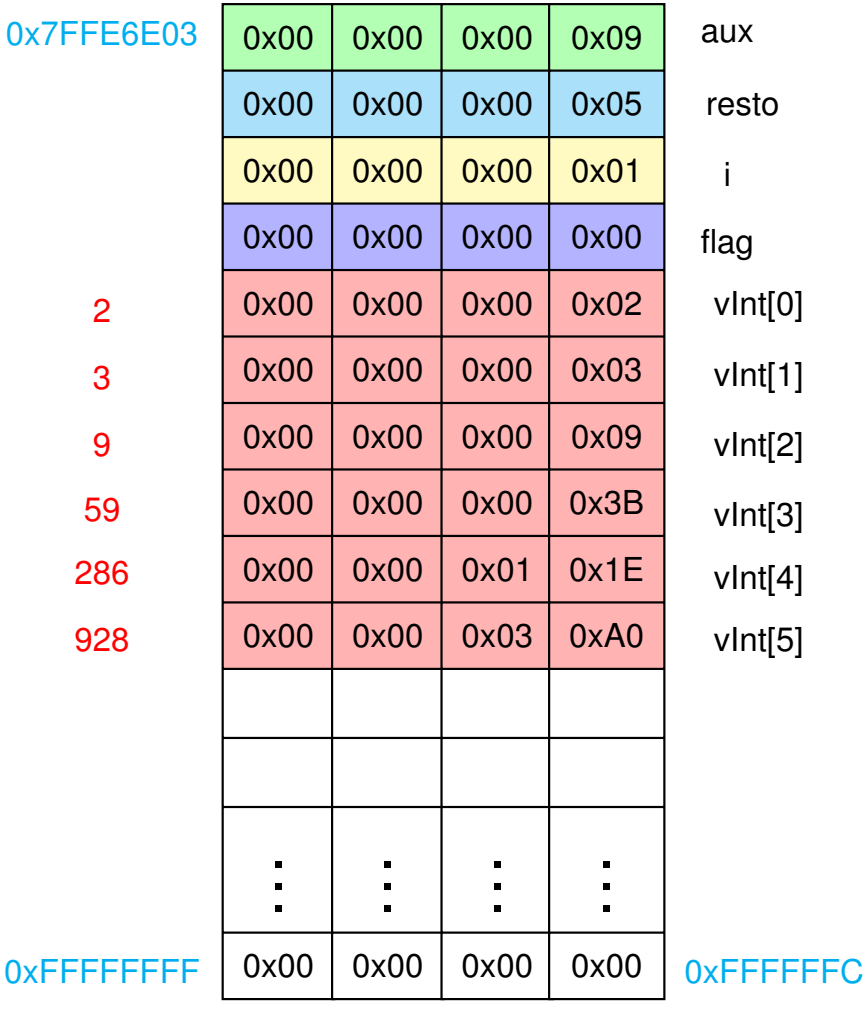
MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

6-5=1

MEMORIA en ARQUITECTURA x86 32-bits



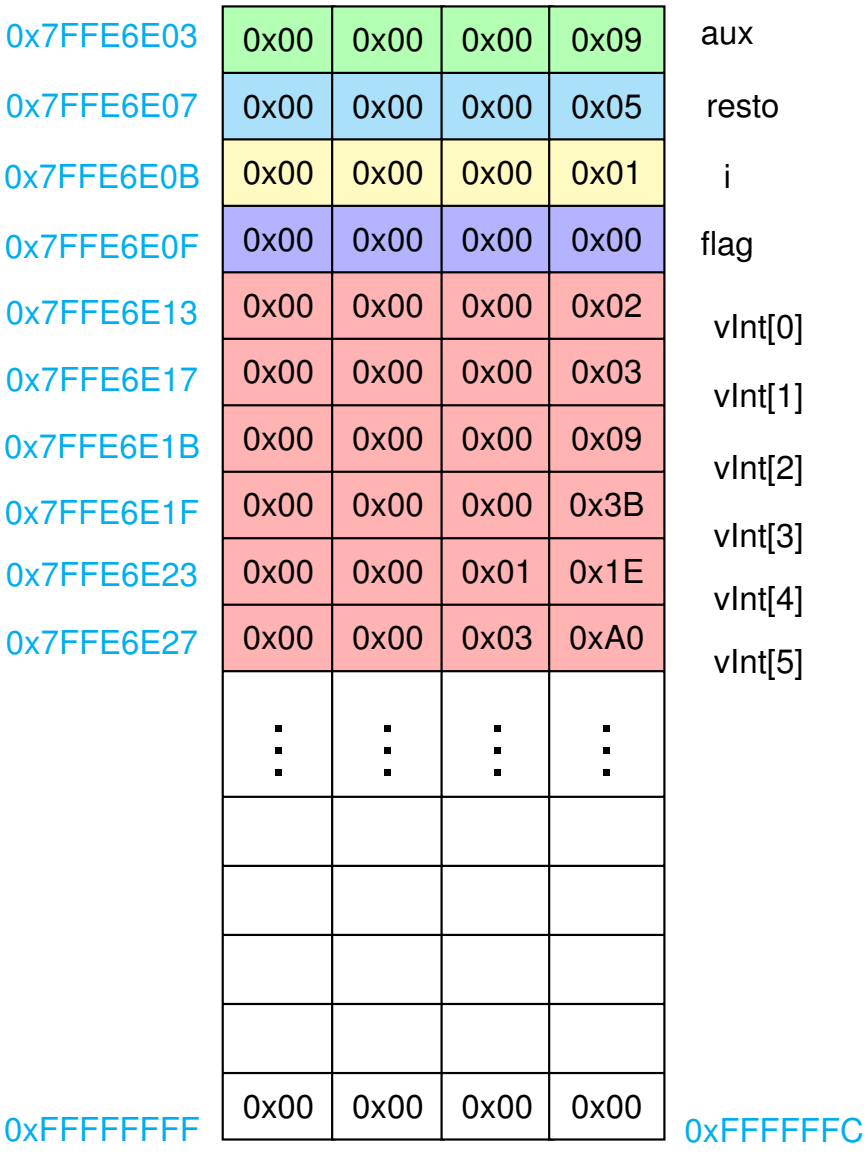

```
1  #include <stdio.h>
2  #define CANT       6
3
4  int main (void)
5  {
6  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
7    resto=0;
8    do
9    {
10     resto++;
11     flag=0;
12     for(i=0;i < (CANT-resto);i++)
13     {
14       if(vInt[i] > vInt[i+1])
15       {
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23 return (0);
24 }
```

Con while(0) sale del loop do-while y termina

Sugerimos ver la continuación de este documento una vez que se tenga los conocimientos de PUNTEROS

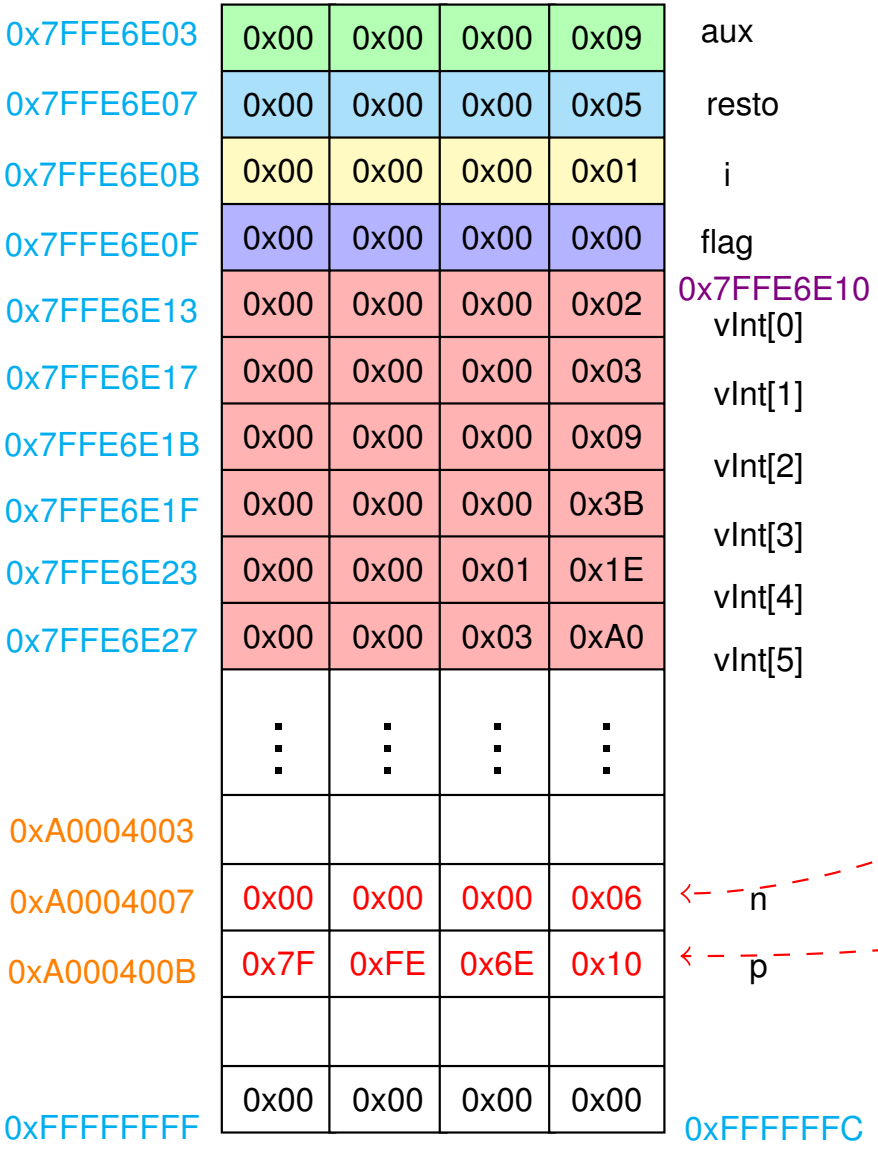
Verificar valores con la función *imprimir*

MEMORIA en ARQUITECTURA x86 32-bits



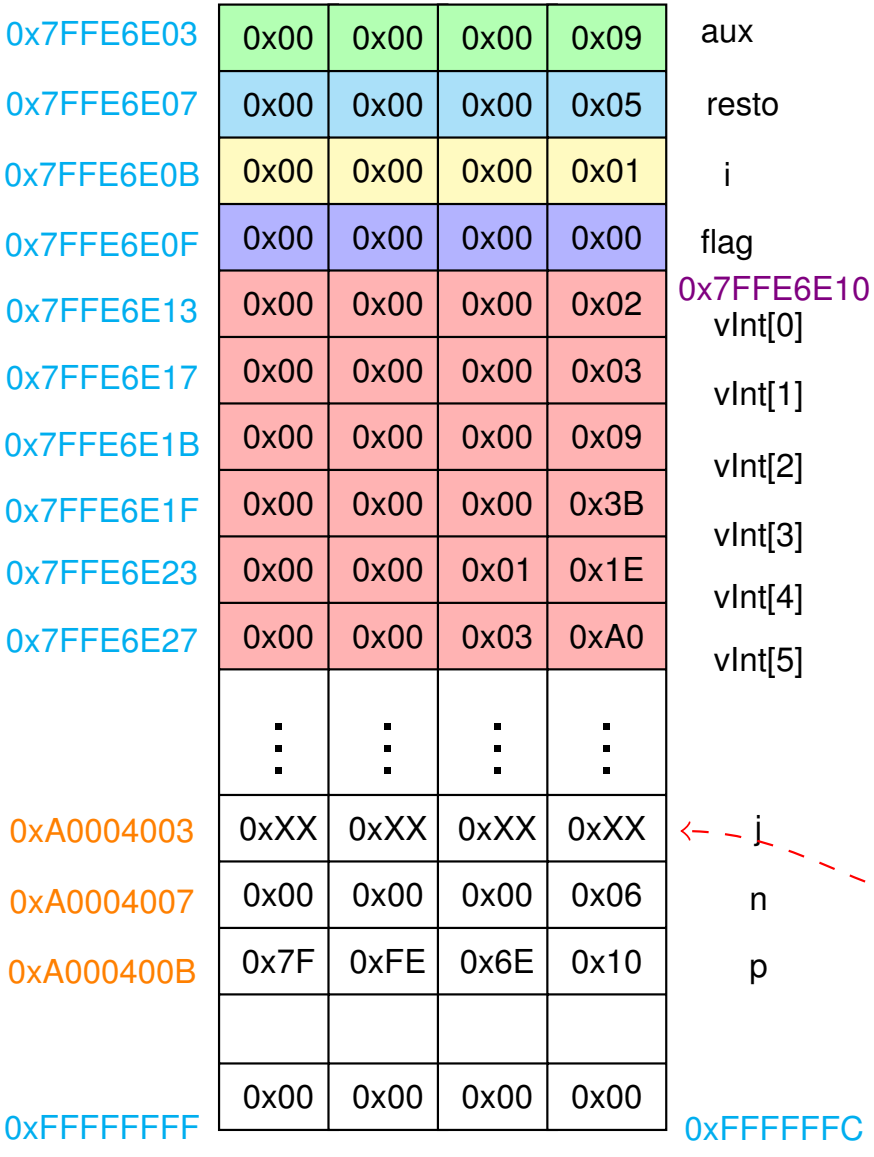

```c
#include <stdio.h>
#define CANT      6

void imprimir(int *p,int n);

int main (void)
{
int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
  imprimir(&vInt[0],CANT);
  resto=0;
  do{
    resto++;
    flag=0;
    for(i=0;i < (CANT-resto);i++){
      if(vInt[i] > vInt[i+1]){
        aux=vInt[i];
        vInt[i]=vInt[i+1];
        vInt[i+1]=aux;
        flag=1;
      }
    }
  }while(flag);
  imprimir(&vInt[0],CANT);
  return (0);
}
void imprimir( int*p,int n)
{
  int j;
  for(j=0;j<n;j++){
   printf("%d \r\n",*(p+j));
  }
}
```

MEMORIA en ARQUITECTURA x86 32-bits



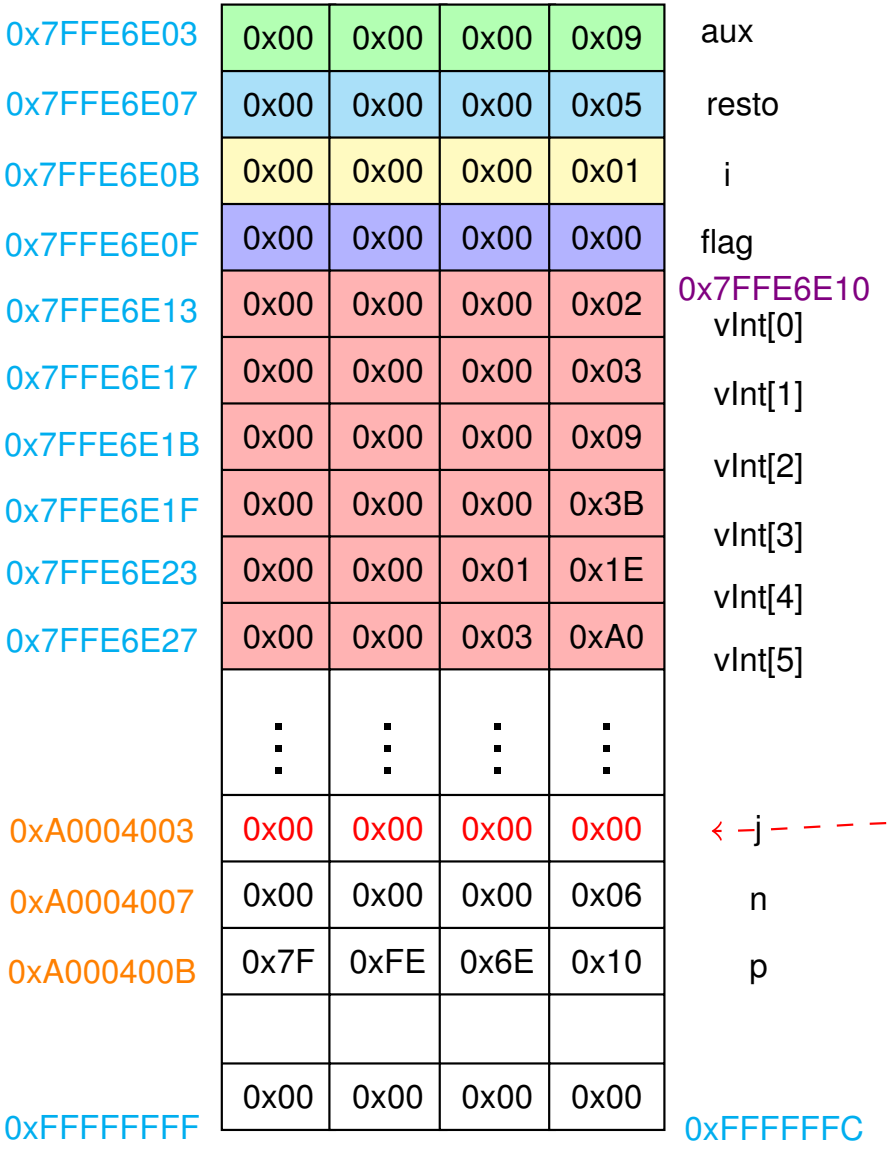

```
1  #include <stdio.h>
2  #define CANT      6
3
4  void imprimir(int *p,int n);
5
6  int main (void)
7  {
8  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
9    imprimir(&vInt[0],CANT);
10   resto=0;
11   do{
12     resto++;
13     flag=0;
14     for(i=0;i < (CANT-resto);i++){
15       if(vInt[i] > vInt[i+1]){
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23   imprimir(&vInt[0],CANT);
24   return (0);
25 }
26 void imprimir( int*p,int n)
27 {
28   int j;
29   for(j=0;j<n;j++){
30     printf("%d \r\n",*(p+j));
31   }
32 }
```

MEMORIA en ARQUITECTURA x86 32-bits



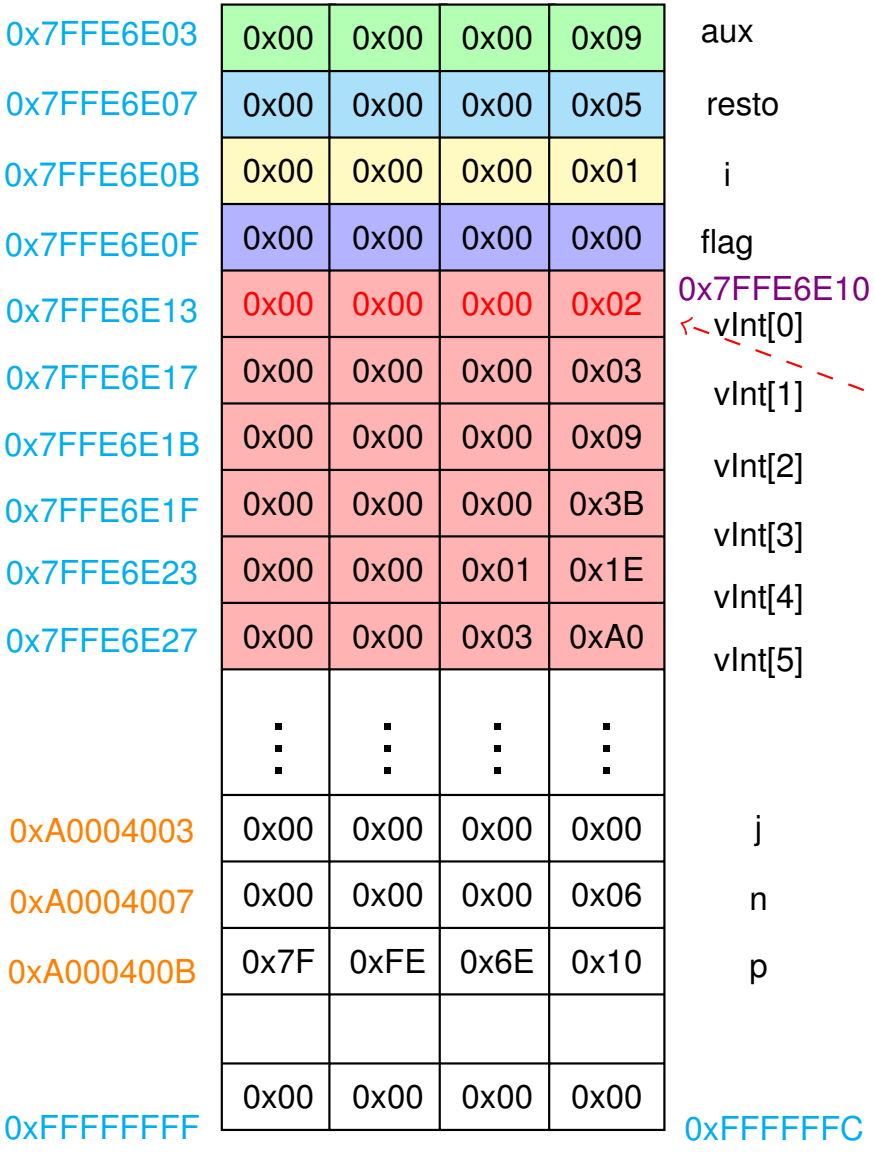

```c
#include <stdio.h>
#define CANT      6

void imprimir(int *p,int n);

int main (void)
{
int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
  imprimir(&vInt[0],CANT);
  resto=0;
  do{
    resto++;
    flag=0;
    for(i=0;i < (CANT-resto);i++){
      if(vInt[i] > vInt[i+1]){
        aux=vInt[i];
        vInt[i]=vInt[i+1];
        vInt[i+1]=aux;
        flag=1;
      }
    }
  }while(flag);
  imprimir(&vInt[0],CANT);
  return (0);
}
void imprimir( int*p,int n)
{
  int j;
  for(j=0;j<n;j++){
   printf("%d \r\n",*(p+j));
  }
}
```

MEMORIA en ARQUITECTURA x86 32-bits



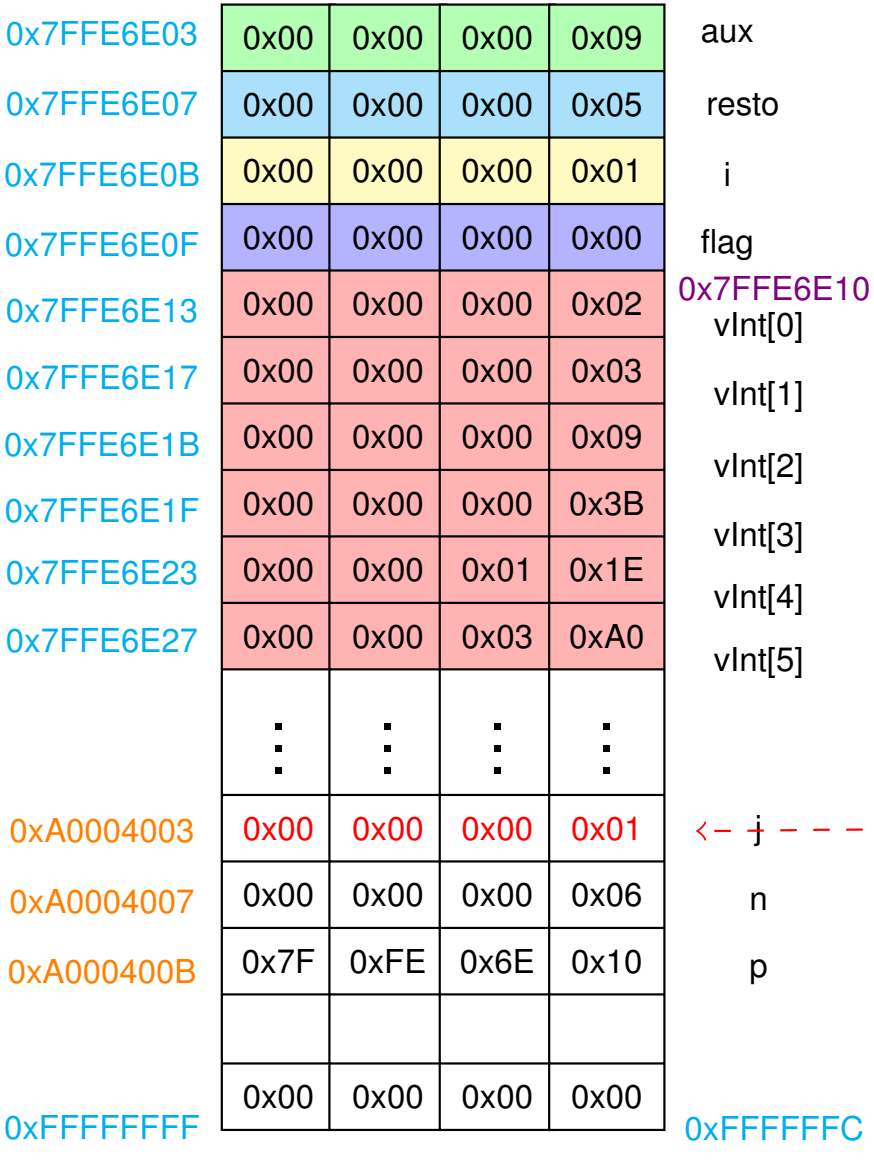

```
1  #include <stdio.h>
2  #define CANT      6
3
4  void imprimir(int *p,int n);
5
6  int main (void)
7  {
8  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
9    imprimir(&vInt[0],CANT);
10   resto=0;
11   do{
12     resto++;
13     flag=0;
14     for(i=0;i < (CANT-resto);i++){
15       if(vInt[i] > vInt[i+1]){
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23   imprimir(&vInt[0],CANT);
24   return (0);
25 }
26 void imprimir( int*p,int n)
27 {
28   int j;
29   for(j=0;j<n;j++){
30    printf("%d \r\n",*(p+j));
31   }
32 }
```

MEMORIA en ARQUITECTURA x86 32-bits



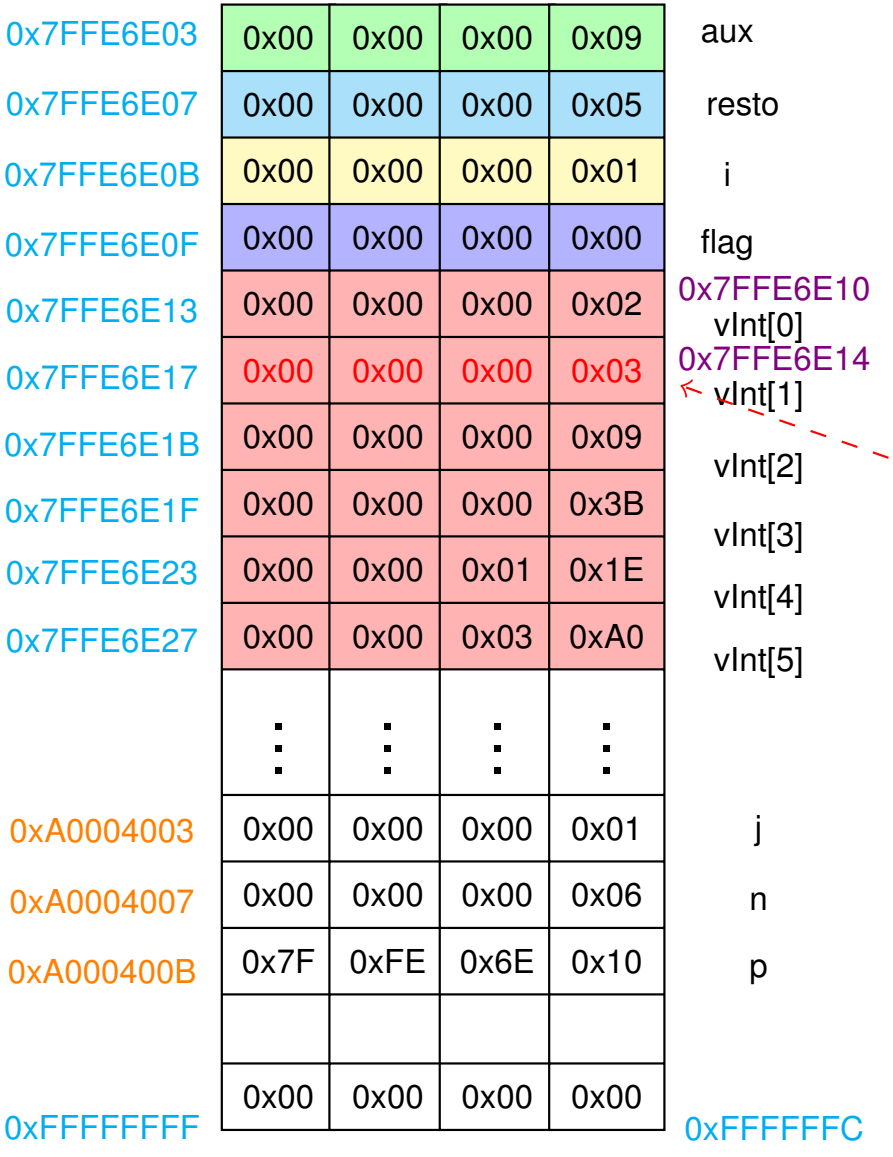

```
1  #include <stdio.h>
2  #define CANT      6
3
4  void imprimir(int *p,int n);
5
6  int main (void)
7  {
8  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
9    imprimir(&vInt[0],CANT);
10   resto=0;
11   do{
12     resto++;
13     flag=0;
14     for(i=0;i < (CANT-resto);i++){
15       if(vInt[i] > vInt[i+1]){
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23   imprimir(&vInt[0],CANT);
24   return (0);
25 }
26 void imprimir( int*p,int n)
27 {
28   int j;
29   for(j=0;j<n;j++){
30    printf("%d \r\n",*(p+j));
31   }
32 }
```

MEMORIA en ARQUITECTURA x86 32-bits



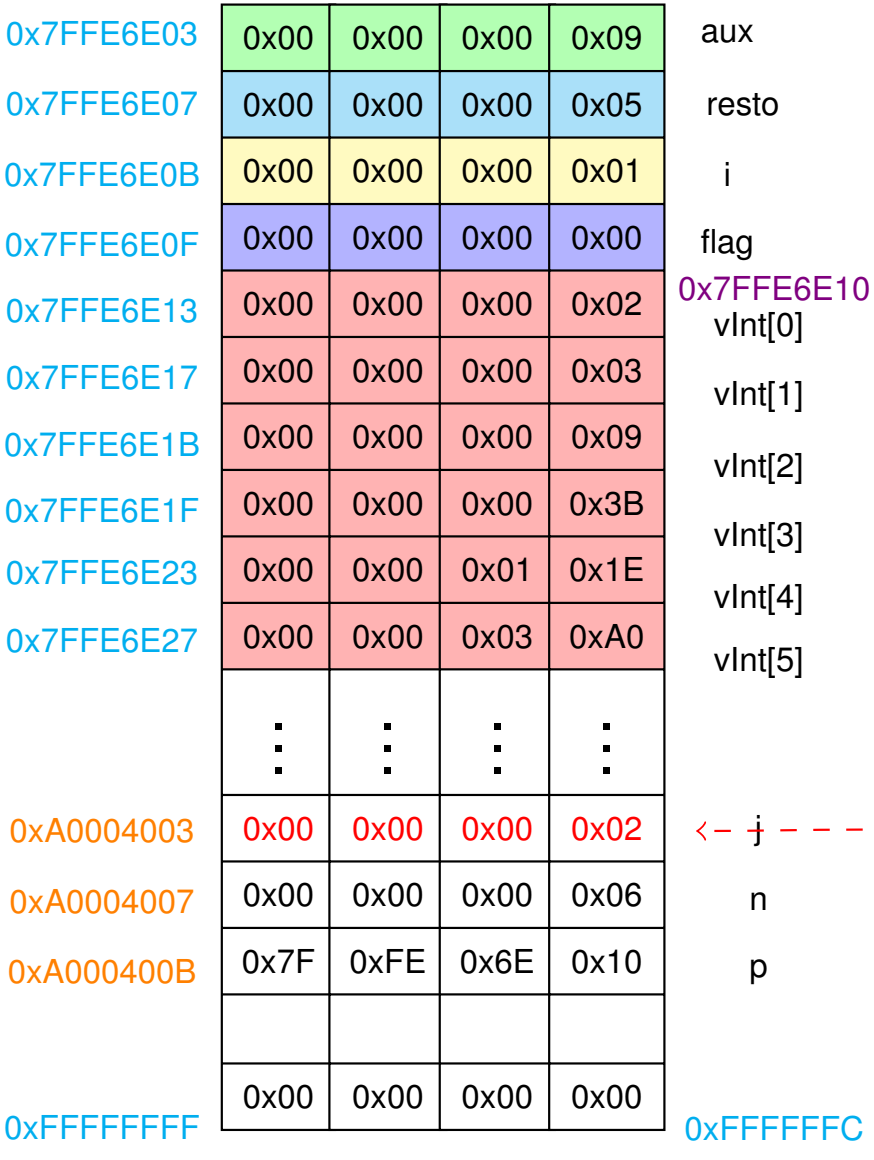

```
1  #include <stdio.h>
2  #define CANT      6
3
4  void imprimir(int *p,int n);
5
6  int main (void)
7  {
8  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
9    imprimir(&vInt[0],CANT);
10   resto=0;
11   do{
12     resto++;
13     flag=0;
14     for(i=0;i < (CANT-resto);i++){
15       if(vInt[i] > vInt[i+1]){
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23   imprimir(&vInt[0],CANT);
24   return (0);
25 }
26 void imprimir( int*p,int n)
27 {
28   int j;
29   for(j=0;j<n;j++){
30    printf("%d \r\n",*(p+j));
31   }
32 }
```

MEMORIA en ARQUITECTURA x86 32-bits



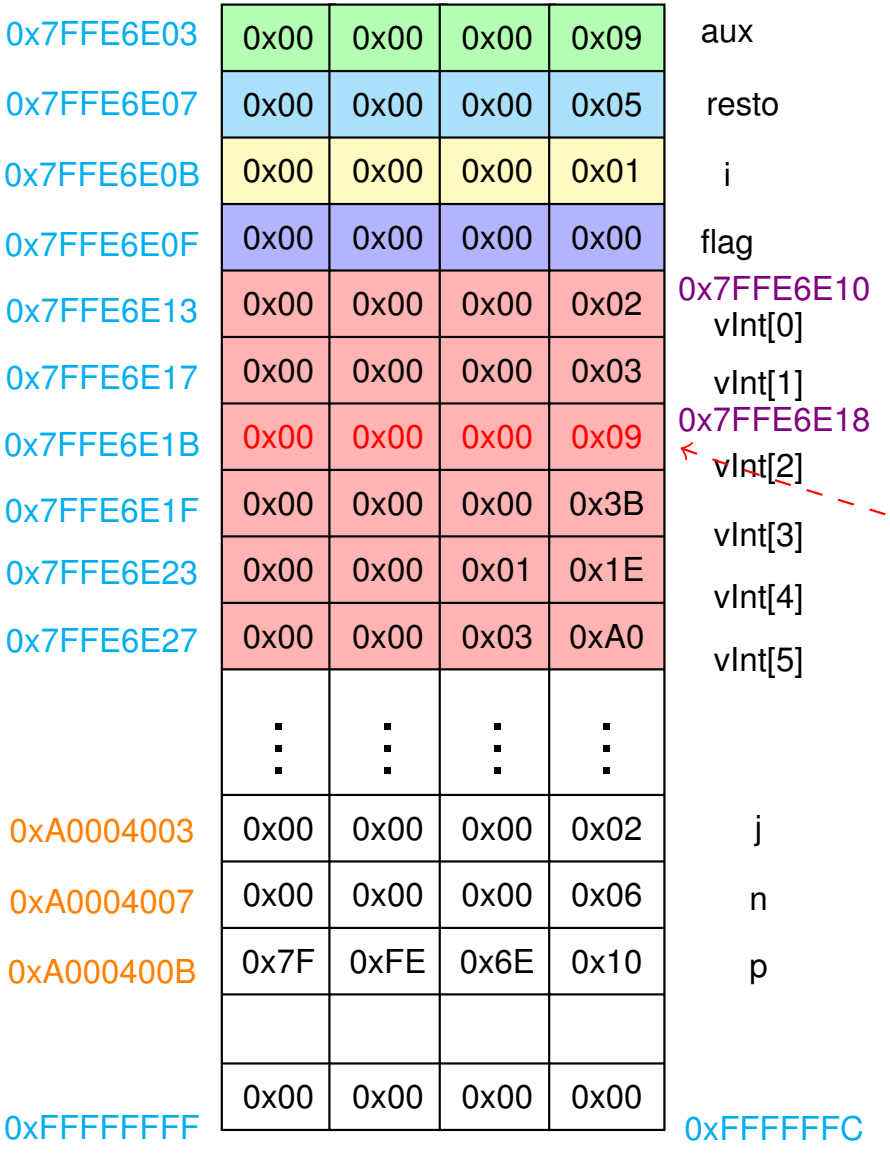

```
1  #include <stdio.h>
2  #define CANT      6
3
4  void imprimir(int *p,int n);
5
6  int main (void)
7  {
8  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
9    imprimir(&vInt[0],CANT);
10   resto=0;
11   do{
12     resto++;
13     flag=0;
14     for(i=0;i < (CANT-resto);i++){
15       if(vInt[i] > vInt[i+1]){
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23   imprimir(&vInt[0],CANT);
24   return (0);
25 }
26 void imprimir( int*p,int n)
27 {
28   int j;
29   for(j=0;j<n;j++){
30    printf("%d \r\n",*(p+j));
31   }
32 }
```

MEMORIA en ARQUITECTURA x86 32-bits



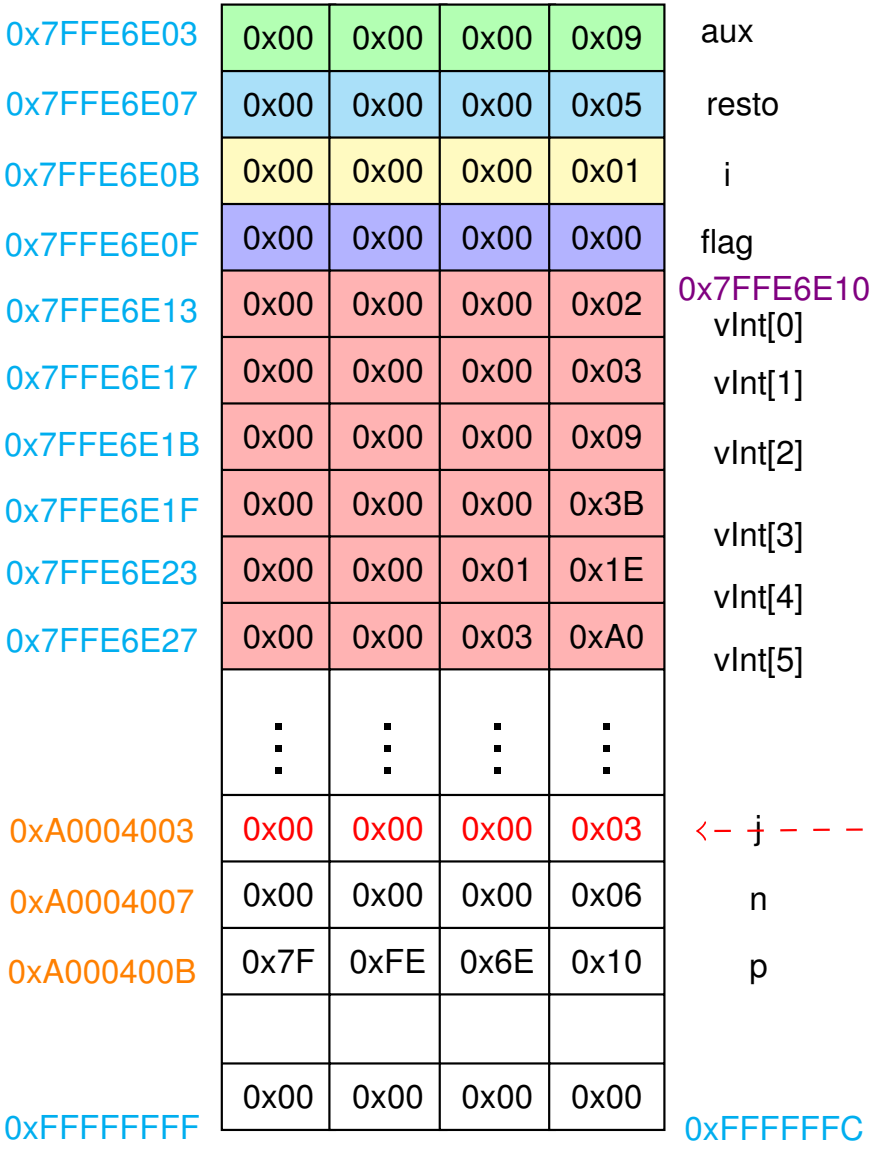

```
1  #include <stdio.h>
2  #define CANT      6
3
4  void imprimir(int *p,int n);
5
6  int main (void)
7  {
8  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
9    imprimir(&vInt[0],CANT);
10   resto=0;
11   do{
12     resto++;
13     flag=0;
14     for(i=0;i < (CANT-resto);i++){
15       if(vInt[i] > vInt[i+1]){
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23   imprimir(&vInt[0],CANT);
24   return (0);
25 }
26 void imprimir( int*p,int n)
27 {
28   int j;
29   for(j=0;j<n;j++){
30    printf("%d \r\n",*(p+j));
31   }
32 }
```

MEMORIA en ARQUITECTURA x86 32-bits



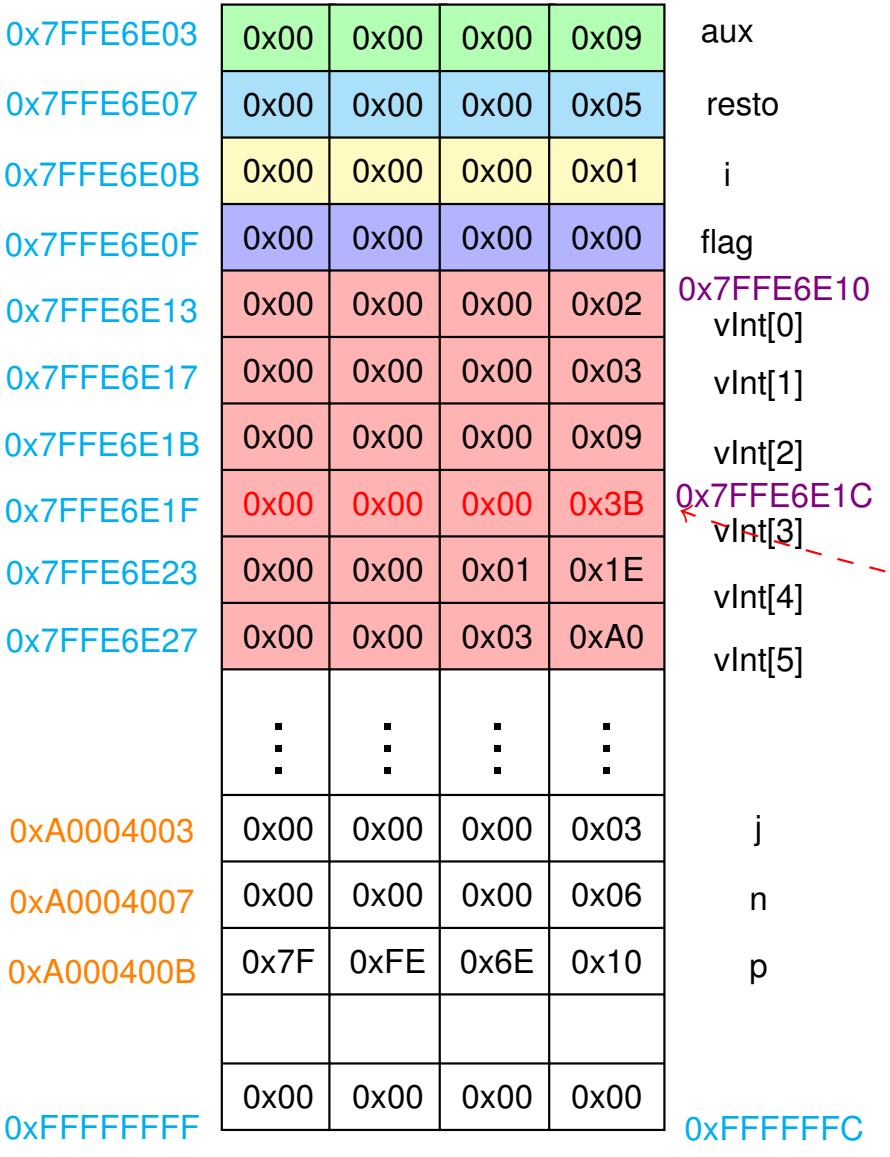

```c
#include <stdio.h>
#define CANT      6

void imprimir(int *p,int n);

int main (void)
{
int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
  imprimir(&vInt[0],CANT);
  resto=0;
  do{
    resto++;
    flag=0;
    for(i=0;i < (CANT-resto);i++){
      if(vInt[i] > vInt[i+1]){
        aux=vInt[i];
        vInt[i]=vInt[i+1];
        vInt[i+1]=aux;
        flag=1;
      }
    }
  }while(flag);
  imprimir(&vInt[0],CANT);
  return (0);
}
void imprimir( int*p,int n)
{
  int j;
  for(j=0;j<n;j++){
   printf("%d \r\n",*(p+j));
  }
}
```

MEMORIA en ARQUITECTURA x86 32-bits



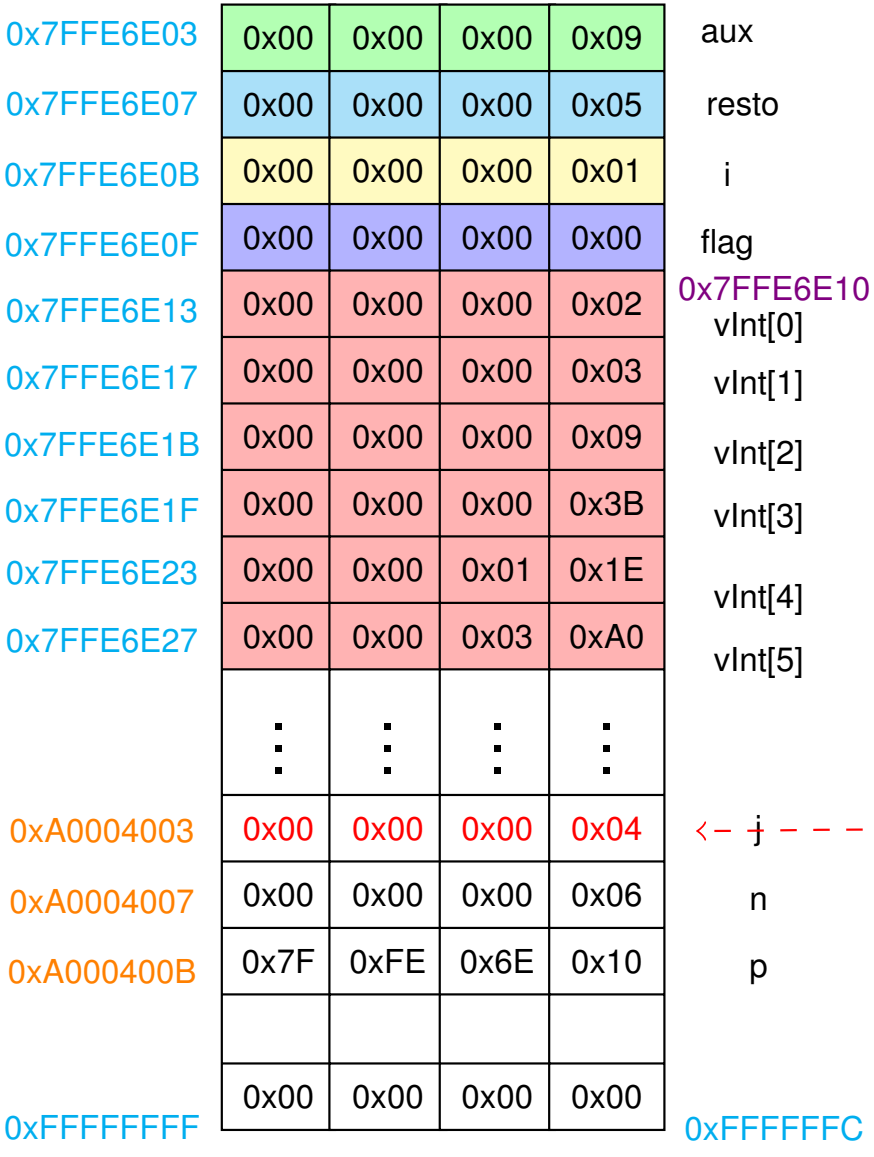

```
1  #include <stdio.h>
2  #define CANT      6
3
4  void imprimir(int *p,int n);
5
6  int main (void)
7  {
8  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
9    imprimir(&vInt[0],CANT);
10   resto=0;
11   do{
12     resto++;
13     flag=0;
14     for(i=0;i < (CANT-resto);i++){
15       if(vInt[i] > vInt[i+1]){
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23   imprimir(&vInt[0],CANT);
24   return (0);
25 }
26 void imprimir( int*p,int n)
27 {
28   int j;
29   for(j=0;j<n;j++){
30    printf("%d \r\n",*(p+j));
31   }
32 }
```

MEMORIA en ARQUITECTURA x86 32-bits



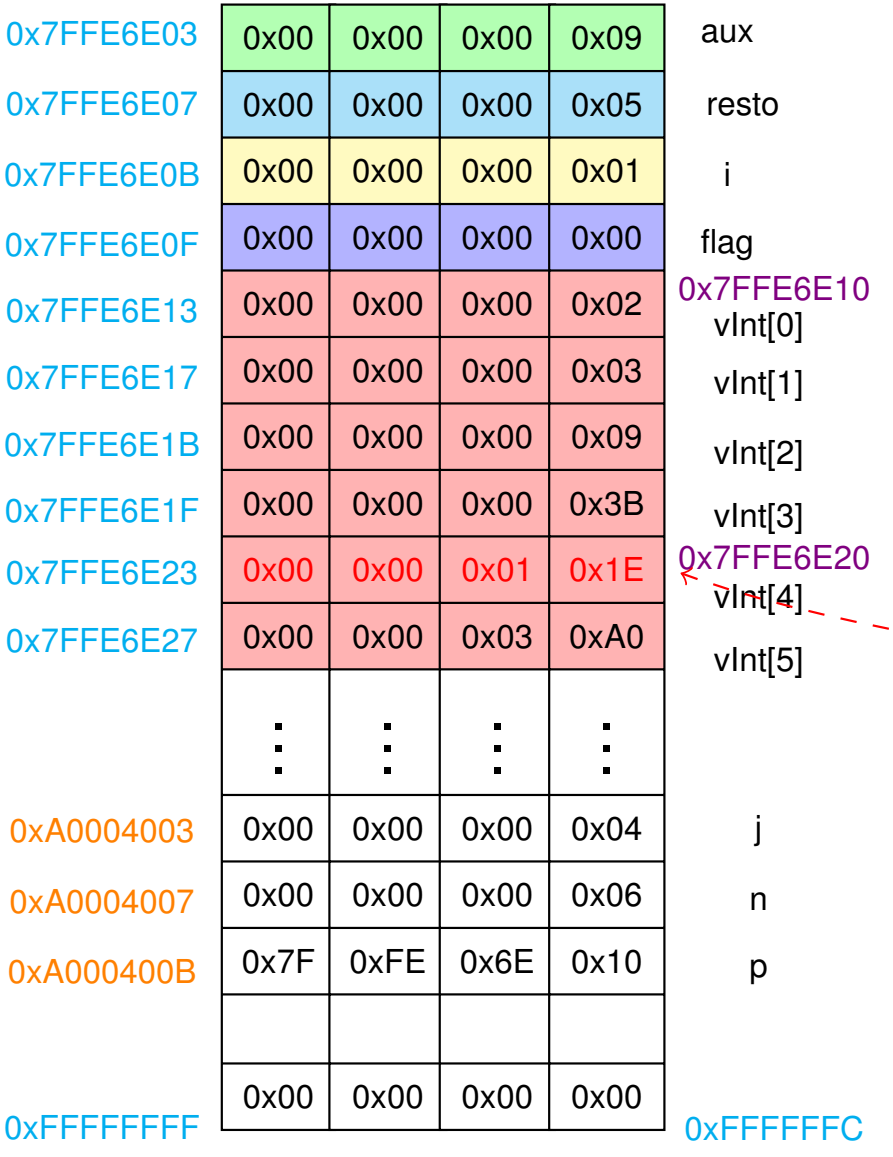

```
1  #include <stdio.h>
2  #define CANT      6
3
4  void imprimir(int *p,int n);
5
6  int main (void)
7  {
8  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
9    imprimir(&vInt[0],CANT);
10   resto=0;
11   do{
12     resto++;
13     flag=0;
14     for(i=0;i < (CANT-resto);i++){
15       if(vInt[i] > vInt[i+1]){
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23   imprimir(&vInt[0],CANT);
24   return (0);
25 }
26 void imprimir( int*p,int n)
27 {
28   int j;
29   for(j=0;j<n;j++){
30    printf("%d \r\n",*(p+j));
31   }
32 }
```

MEMORIA en ARQUITECTURA x86 32-bits



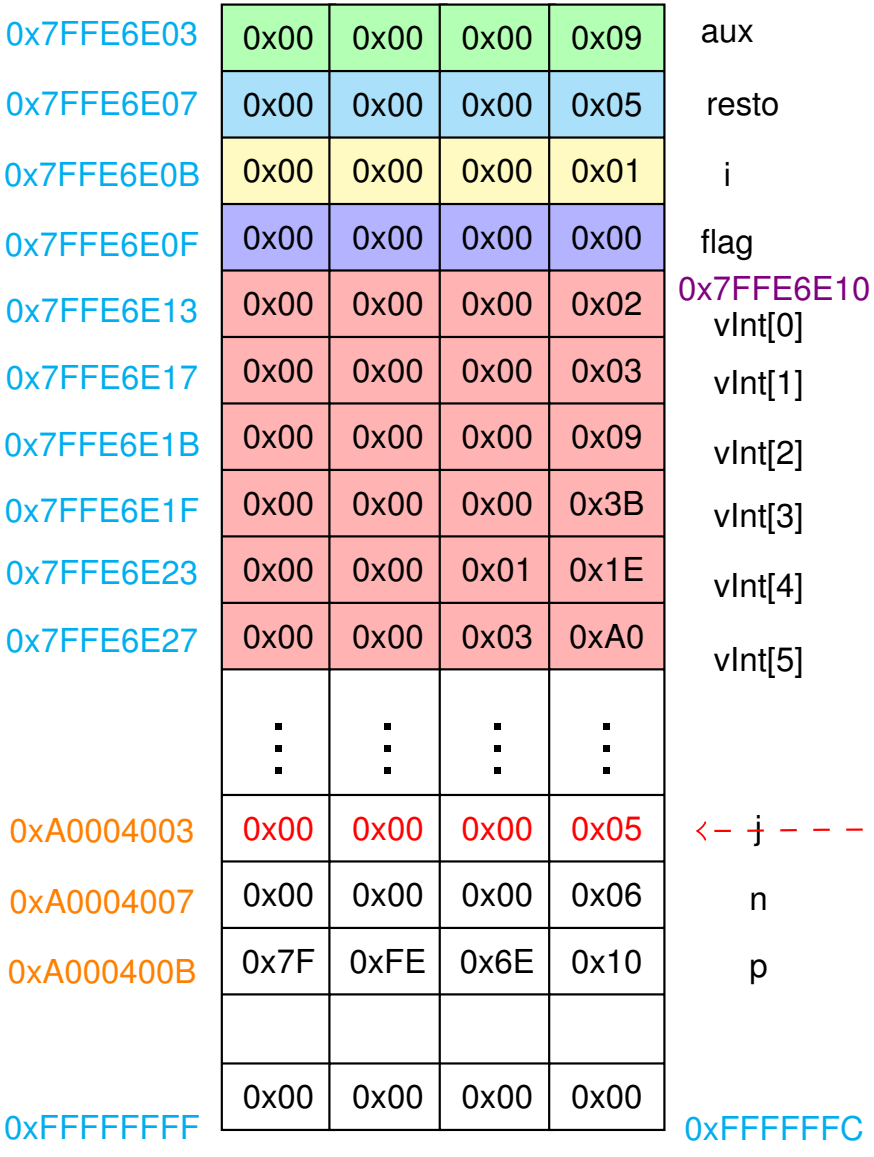

```
1  #include <stdio.h>
2  #define CANT      6
3
4  void imprimir(int *p,int n);
5
6  int main (void)
7  {
8  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
9    imprimir(&vInt[0],CANT);
10   resto=0;
11   do{
12     resto++;
13     flag=0;
14     for(i=0;i < (CANT-resto);i++){
15       if(vInt[i] > vInt[i+1]){
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23   imprimir(&vInt[0],CANT);
24   return (0);
25 }
26 void imprimir( int*p,int n)
27 {
28   int j;
29   for(j=0;j<n;j++){
30    printf("%d \r\n",*(p+j));
31   }
32 }
```

MEMORIA en ARQUITECTURA x86 32-bits



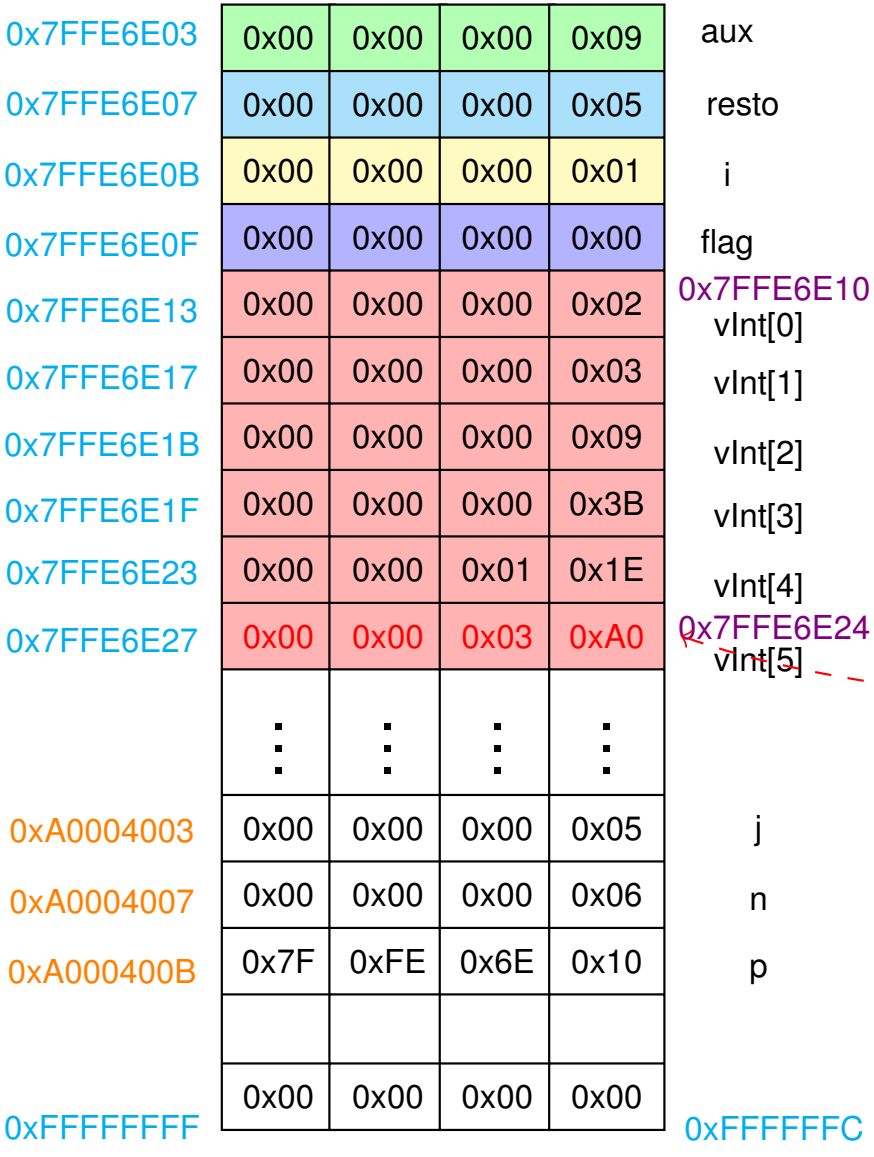

```
1  #include <stdio.h>
2  #define CANT      6
3
4  void imprimir(int *p,int n);
5
6  int main (void)
7  {
8  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
9    imprimir(&vInt[0],CANT);
10   resto=0;
11   do{
12     resto++;
13     flag=0;
14     for(i=0;i < (CANT-resto);i++){
15       if(vInt[i] > vInt[i+1]){
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23   imprimir(&vInt[0],CANT);
24   return (0);
25 }
26 void imprimir( int*p,int n)
27 {
28   int j;
29   for(j=0;j<n;j++){
30    printf("%d \r\n",*(p+j));
31   }
32 }
```

MEMORIA en ARQUITECTURA x86 32-bits



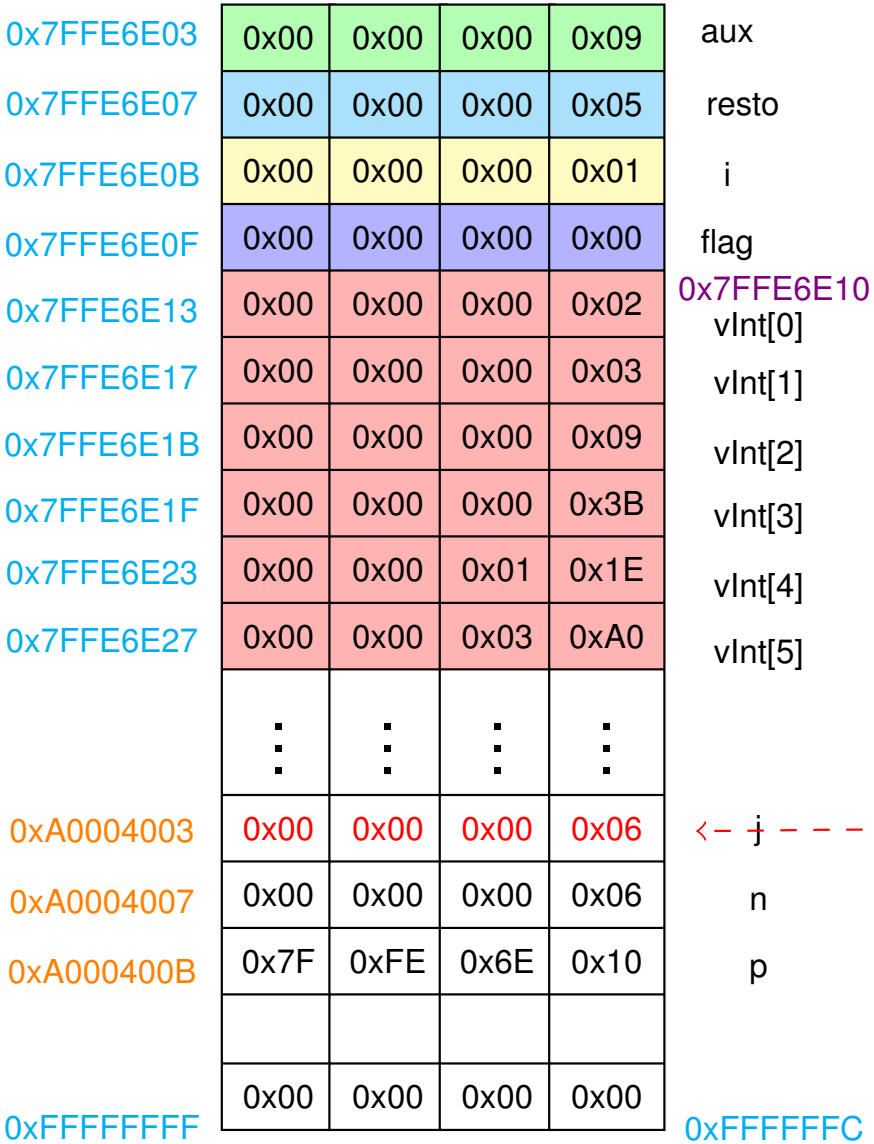

```
1  #include <stdio.h>
2  #define CANT      6
3
4  void imprimir(int *p,int n);
5
6  int main (void)
7  {
8  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
9    imprimir(&vInt[0],CANT);
10   resto=0;
11   do{
12     resto++;
13     flag=0;
14     for(i=0;i < (CANT-resto);i++){
15       if(vInt[i] > vInt[i+1]){
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23   imprimir(&vInt[0],CANT);
24   return (0);
25 }
26 void imprimir( int*p,int n)
27 {
28   int j;
29   for(j=0;j<n;j++){
30    printf("%d \r\n",*(p+j));
31   }
32 }
```

MEMORIA en ARQUITECTURA x86 32-bits

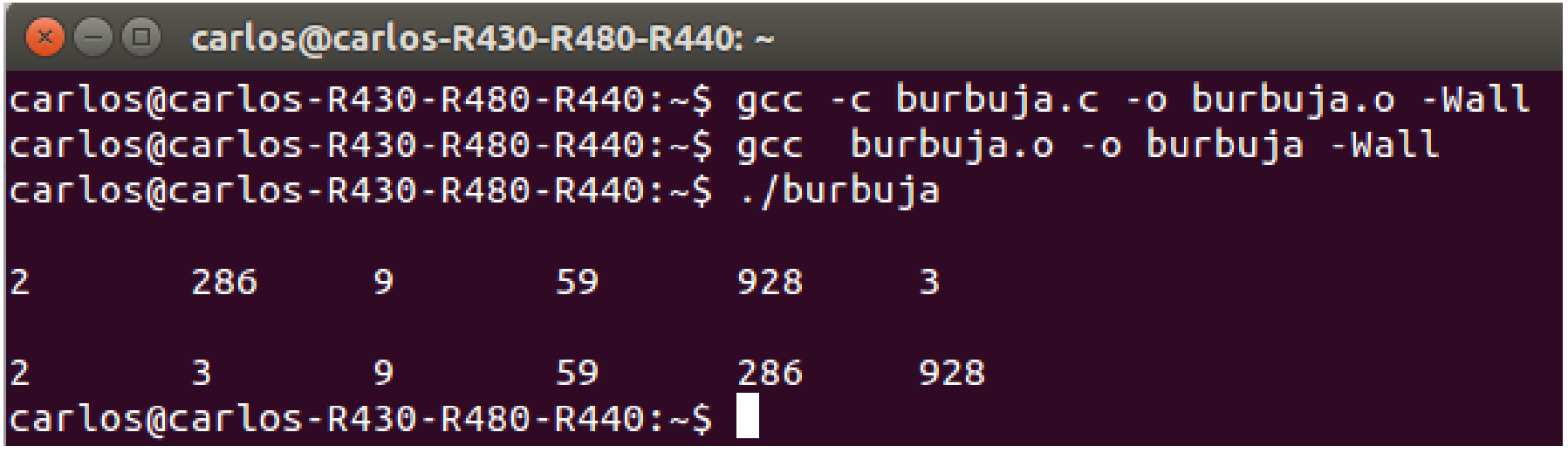



```
1  #include <stdio.h>
2  #define CANT      6
3
4  void imprimir(int *p,int n);
5
6  int main (void)
7  {
8  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
9    imprimir(&vInt[0],CANT);
10   resto=0;
11   do{
12     resto++;
13     flag=0;
14     for(i=0;i < (CANT-resto);i++){
15       if(vInt[i] > vInt[i+1]){
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23   imprimir(&vInt[0],CANT);
24   return (0);
25 }
26 void imprimir( int*p,int n)
27 {
28   int j;
29   for(j=0;j<n;j++){
30    printf("%d \r\n",*(p+j));
31   }
32 }
```

MEMORIA en ARQUITECTURA x86 32-bits




```
1  #include <stdio.h>
2  #define CANT      6
3
4  void imprimir(int *p,int n);
5
6  int main (void)
7  {
8  int aux,resto,i,flag,vInt[CANT]={2,286,9,59,928,3};
9    imprimir(&vInt[0],CANT);
10   resto=0;
11   do{
12     resto++;
13     flag=0;
14     for(i=0;i < (CANT-resto);i++){
15       if(vInt[i] > vInt[i+1]){
16         aux=vInt[i];
17         vInt[i]=vInt[i+1];
18         vInt[i+1]=aux;
19         flag=1;
20       }
21     }
22   }while(flag);
23   imprimir(&vInt[0],CANT);
24   return (0);
25 }
26 void imprimir( int*p,int n)
27 {
28   int j;
29   for(j=0;j<n;j++){
30    printf("%d \r\n",*(p+j));
31   }
32 }
```



- compila con ***gcc -c burbuja.c -o burbuja.o -Wall***
- Linkea con ***gcc burbuja.o -o burbuja -Wall***
- Ejecuta con ***./burbuja***
- Primero imprime en la consola los valores desordenados
- Al finalizar el ordenamiento también los imprime en la consola