Programación Paralela y Distribuida
Clase 01 - Aritmética de Punteros

John Sanabria - john.sanabria@correounivalle.edu.co

Aritmética de Punteros en C

- La memoria del computador se puede percibir como un arreglo lineal de almacenamiento de datos y código
- Cada dirección de memoria apunta a un byte

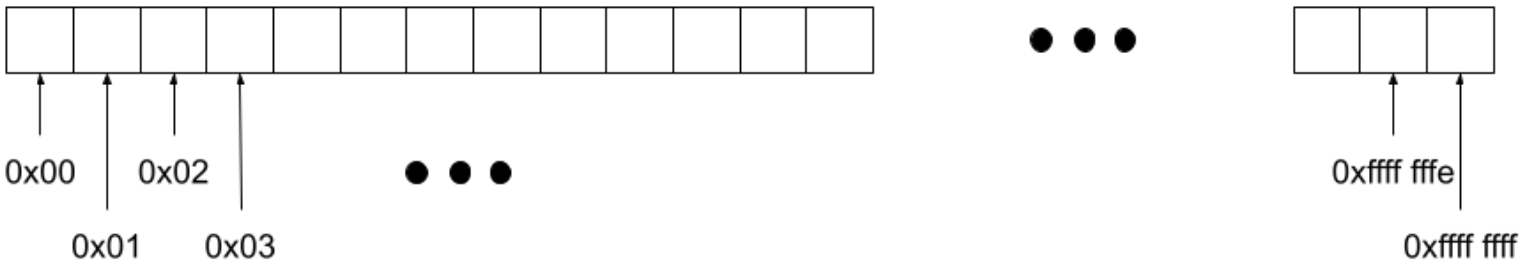

Figure 1: Memoria con direcciones de 0x0000 0000 hasta 0xffff ffff - 2^{32}

Un comentario frente a la numeración en hardware

- La forma como se identifican las direcciones de memoria es a través de la **notación hexadecimal**. Ejemplo: $0xF9E34518ACE00BD2$. La $0x$ denota que es un número en hexadecimal

- Decimal

0
1
2
3
4
5
6
7
8
9
10

- Hexadecimal

0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F
10

Ejercicios

- Cuanto da la suma de $0xABCD1234$ con $0x67892345$?
- Cuanto da la suma de $0xABCD1234$ con 67892345?

Región de memoria asignada para contener a un vector de 12 **enteros**

- Observe de posición a posición hay un incremento de 4, 0x1234 → 0x1238, 0x1248 → 0x124C, 0x125C → 0x1260
- Este incremento de 4 en 4 se da porque **cada posición de memoria apunta a un dato entero** el cual ocupa 4 bytes

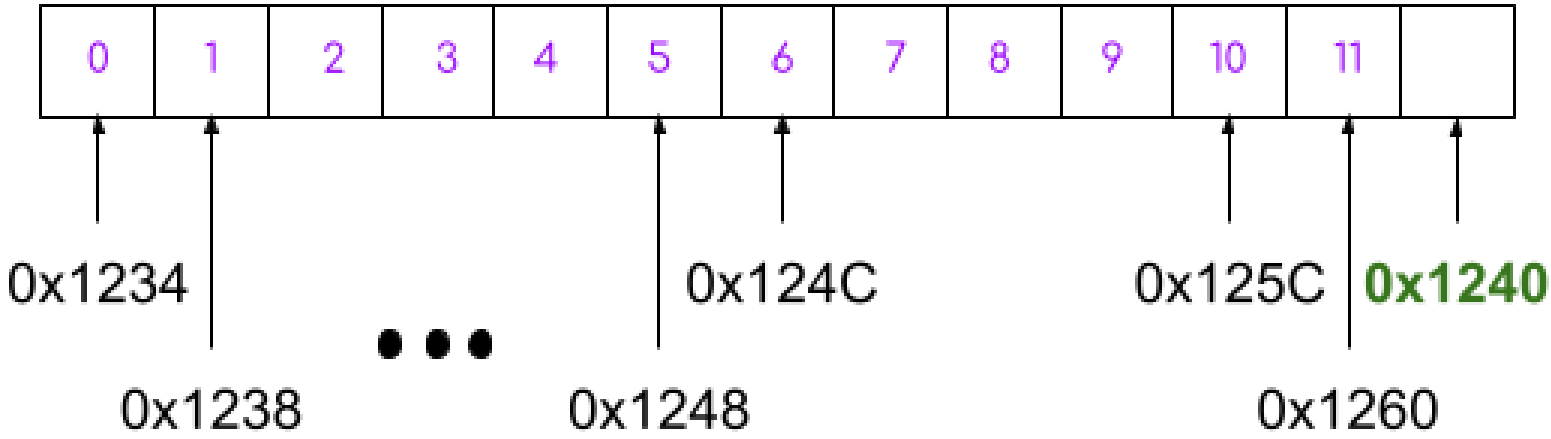

Figure 2: Arreglo de 12 elementos que comienza en dirección de memoria 0x1234

Definición y uso de punteros en C

```c
#include <stdio.h>

int main() {
  int vector[] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 };
  int *ptr;

  ptr = &vector[0]; // es equivalente 'ptr = vector;'
  printf("Direccion del puntero %p\n",ptr);
  printf("Direccion base de vector %p\n",vector);
  printf("Direccion base de vector %p\n",&vector[0]);
  printf("Valor en esta direccion %d\n",*ptr);
  printf("Valor en esta direccion %d\n",ptr[0]);
  printf("Valor en esta direccion %d\n",vector[0]);

  return 0;
}
```

Definición y uso de punteros en C (2)

```c
#include <stdio.h>

int main() {
  int vector[] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 };
  int *ptr;

  ptr = &vector[0];
  printf("Valor en posicion 1 %d\n",ptr[1]); // '1'
  ptr = ptr + 1;
  printf("Direccion de memoria %p y su valor %d\n",ptr, *ptr); // '1'
  ptr = ptr + 2;
  printf("Direccion de memoria %p y su valor %d\n",ptr, *ptr); // '3'
  printf("Direccion de memoria %p y su valor %d\n",ptr, ptr[0]); // '3'
  printf("Direccion de memoria %p y su valor %d\n",ptr, ptr[2]); // '5'

  return 0;
}
```

Visualización de lo presentado anteriormente

Figure 3: Apuntador sin inicializar

Visualización de lo presentado anteriormente (2)

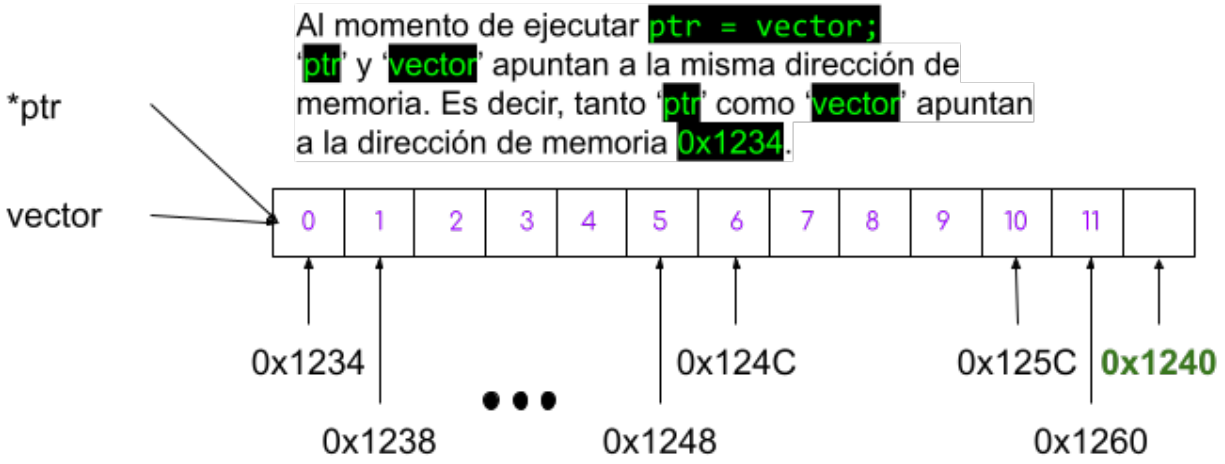

Figure 4: Apuntador sin inicializar

Visualización de lo presentado anteriormente (3)

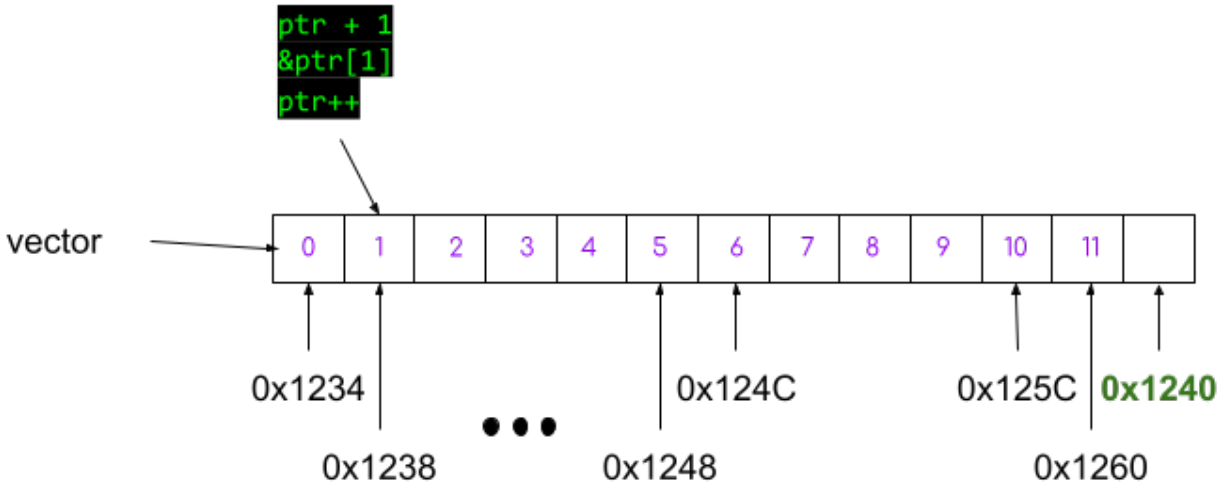

Figure 5: Apuntador sin inicializar

Arreglo visto como una matriz

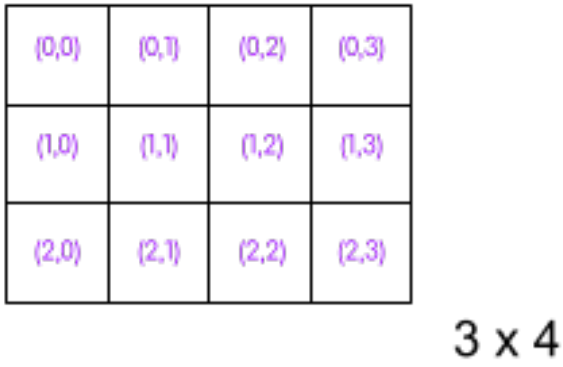

Figure 6: Matriz de 3 x 4

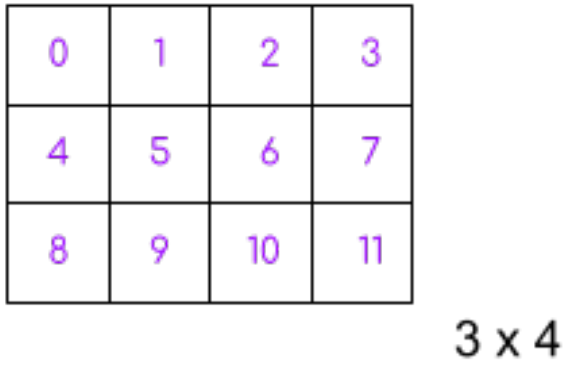

Figure 7: Matriz de 3 x 4

Indique la fórmula para llegar a la posición (i,j) sabiendo que la matriz es de m x n

- ¿Cómo acceder a la posición `(2,1)` a través de `ptr` usando aritmética de punteros?
- ¿Cómo llegó al valor de 9 dada la posición `(2,1)` sabiendo que la matriz es de dimensión 3 `x` 4?

Veamos algo de código en C

Notebook en Google Colab - Aritmética de Punteros