

# ECE264: Advanced C Programming

Summer 2019

Week 2: Addresses, Pointers, Pointer Arithmetic

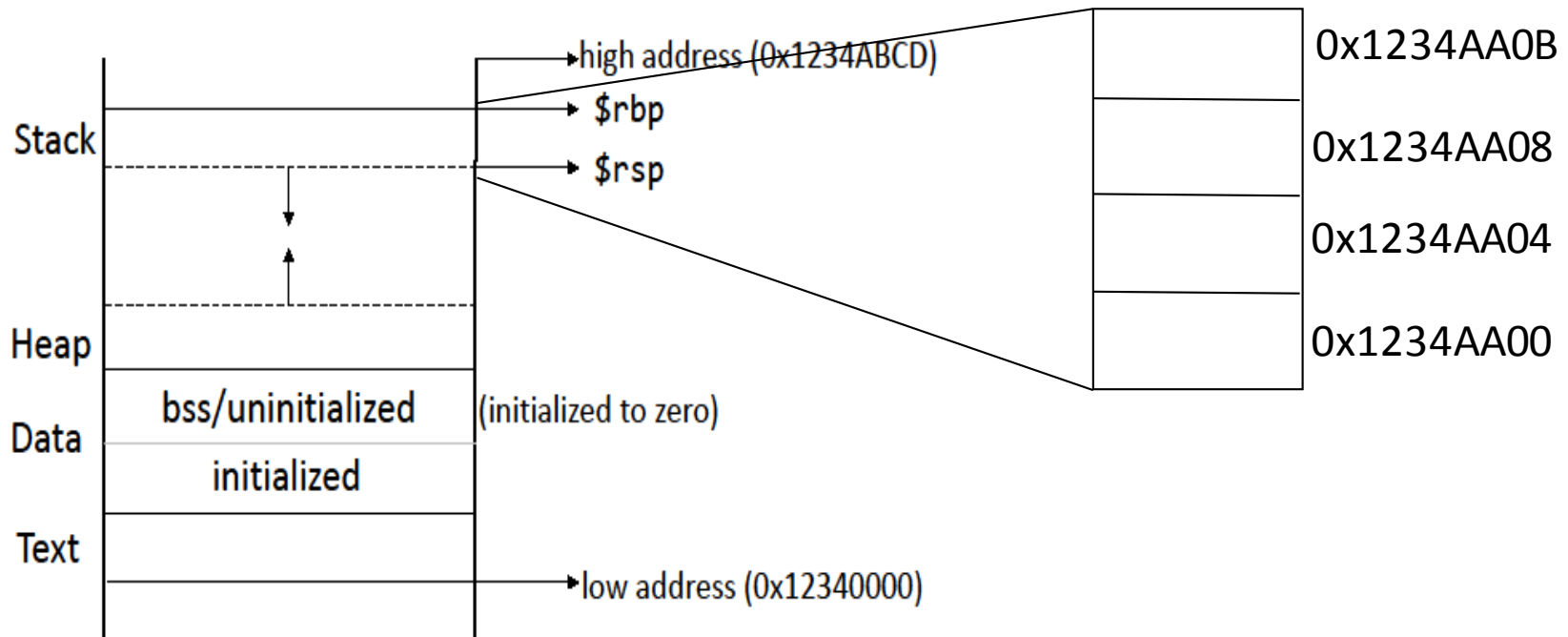
# Addresses

- Humans are not good at remembering numerical addresses.
  - What are the GPS coordinates (latitude and longitude) of your residence?
- Addresses in computer programs are just numbers.

# Addresses

- Addresses in computer programs identify memory locations.
- Computer programs think and live in terms of memory locations.

# Program Memory Layout - Revisited



- Every memory location is a box holding data
- Each box has an address

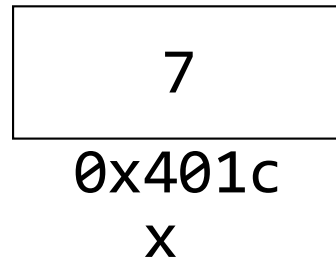
# Addresses

- A program navigates by visiting one address after another.
- We (humans) choose convenient ways to identify addresses so that we can give directions to a program
  - Variables

# Addresses - Handles

- What is a variable?
  - Its just a handle to an address / program memory location

- `int x = 7;`



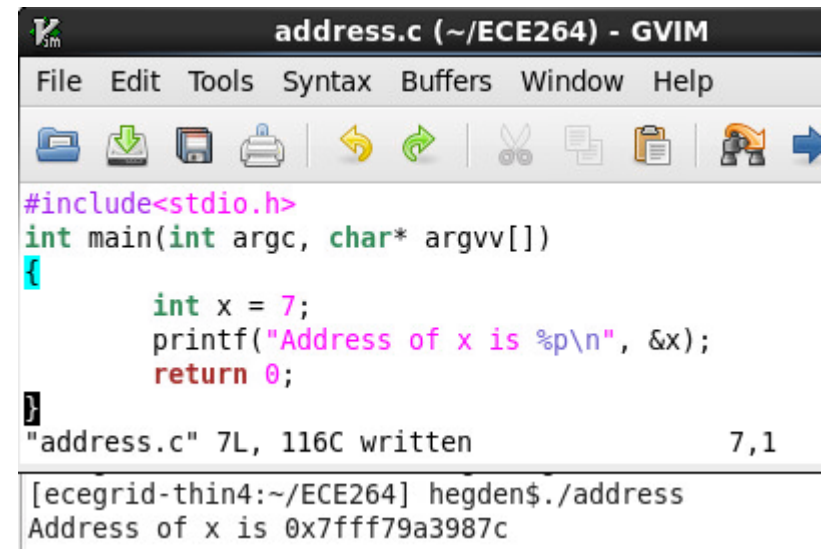
- Read x => Read the content at address 0x401C
- Write x=> Write at address 0x401C

# Addresses - Visualizing

- The *address of* (&) operator fetches a variable's address in C.
- &x would return the address 0x401C.
- Format specifier 'p' :

```
printf("%p\n",&x)
```

prints the Hexadecimal  
address of x



The screenshot shows a Gvim window titled 'address.c (~/ECE264) - Gvim'. The window contains a C program that includes `<stdio.h>` and defines a `main` function. Inside `main`, an integer `x` is set to 7, and `printf` is used to print the address of `x` using the format specifier `%p`. The program then returns 0. Below the code editor, a status line indicates the file size: '"address.c" 7L, 116C written' and the cursor position '7,1'. At the bottom, a terminal window shows the command `./address` being executed, resulting in the output: `Address of x is 0x7fff79a3987c`.

```
address.c (~/ECE264) - Gvim
File Edit Tools Syntax Buffers Window Help

#include<stdio.h>
int main(int argc, char* argv[])
{
    int x = 7;
    printf("Address of x is %p\n", &x);
    return 0;
}

"address.c" 7L, 116C written 7,1

[eccegrid-thin4:~/ECE264] hegden$ ./address
Address of x is 0x7fff79a3987c
```

# Pointers

- Pointer is a data type that *holds an address*.

`<type>* <pointer_name>;`

We read it as “**pointer to** `<type>`”

- Example:

- `int* p;`

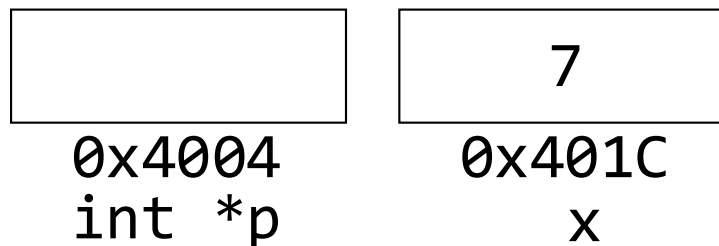
is a variable named `p` whose type is pointer to `int`  
OR `p` is an integer pointer

Note that the variable declared is `p`, *not* `*p`



# Pointers

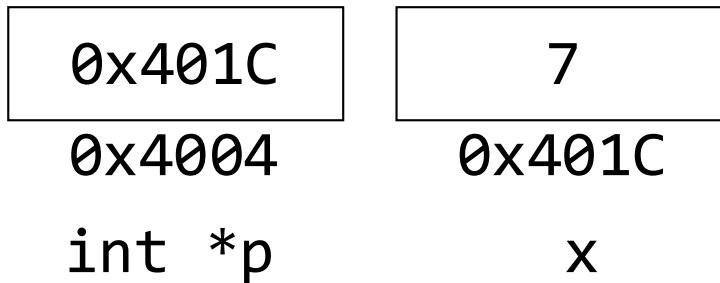
- A pointer always stores an address
- `<type>` of the pointer tells us what kind of data is stored at that address
- Example:
  - `int* p; //declares a pointer variable p holding an address, which can store an integer.`
- Remember p is a variable and all variables are just names identifying addresses. E.g.



# Initializing Pointers

- `int* p=&x;`

//p holds the address of a memory location that stores an integer.



- We say `p` *points to* `x`