Parallel Computing with GPUs

Memory
Part 2 – Advanced use of Pointers

The University Of Sheffield.

Dr Paul Richmond
http://paulrichmond.shef.ac.uk/teaching/COM4521/

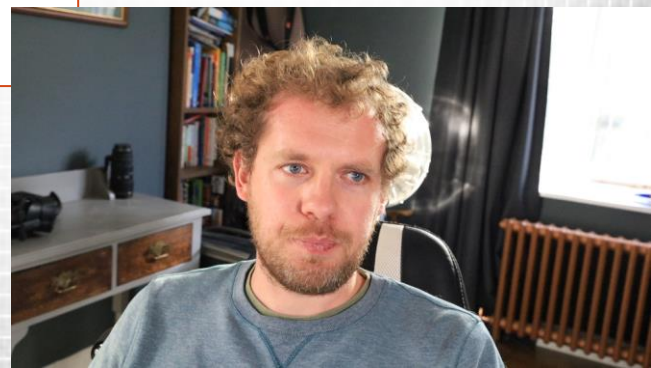
This Lecture (learning objectives)

- Advanced use of Pointers
  - Identify a general purpose pointer
  - Determine the endianness of a computing system
  - Interpret advanced pointer declarations
  - Recognise function pointers and determine where they may be used

General Purpose Pointer

- A General purpose pointer can be defined using `void` type
  - A void type can not be dereferenced
  - Carefull: Arithmetic on a void pointer will increment/decrement by 1 byte
    - Even if it points to a 4 byte data type (e.g. int)

```
void *p;
char c;
int i;
float f;
p = &c;  // ptr has address of character data
p = &i;  // ptr has address of integer data
p = &f;  // ptr has address of float data
```

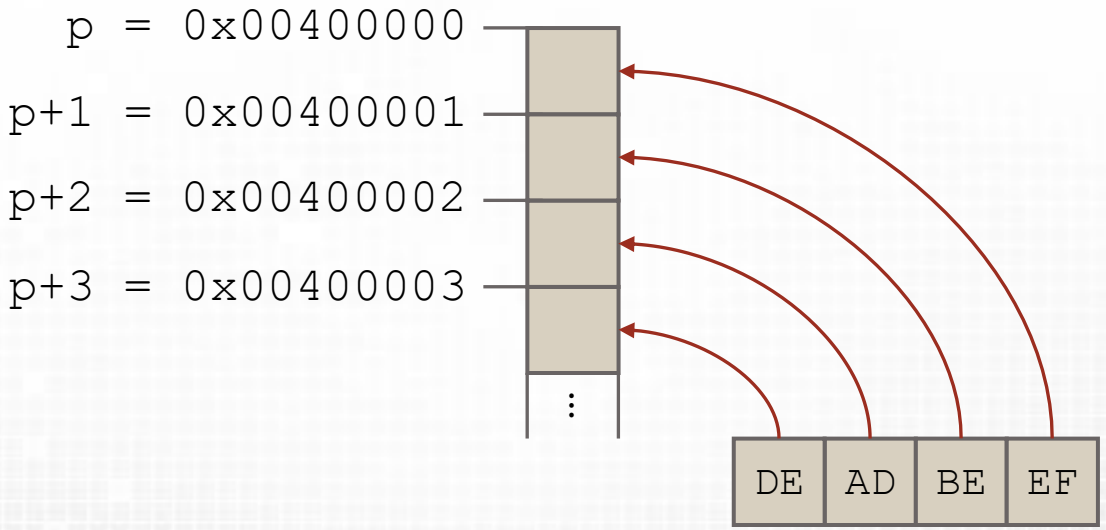


Endianness

- X86 uses little endian format
  - Memory is stored from least significant byte stored at the **lowest** memory

```
unsigned int a = 0xDEADBEEF;
char* p;
p = (char*)&a; //Note explicit cast
printf("0x%08X, 0x%08X, 0x%08X, 0x%08X\n", p, p+1, p+2, p+3);
printf("0x%02X, 0x%02X, 0x%02X, 0x%02X\n", *p, *(p+1), *(p+2), *(p+3));
```

```
0x00400000, 0x00400001, 0x00400002, 0x00400003
0xEF, 0xBE, 0xAD, 0xDE
```



Endianness

```
int a[] = {0x12345678, 0x0000000, 0xDEADBEEF, 0xFFFFFFFF};
```

Endianess is very stange without an example

ssenaidnE si yrev egnats tuohtiw na elpmaxe

Pointers to pointers

- Consider the following
  - `int a[10][20]`
  - `int *b[10]`
- a is a two-dimensional array
  - `200` int sized locations are reserved in memory
- b is single dimensional array of pointers
  - `10` pointers to integers are reserved
  - `B[?]` must be initialised *(or allocated later in this lecture)*
  - The pointers in `b` may be initialised to arrays of different length

```
char names[][10] = {"Paul", "Bob", "Emma", "Jim", "Kathryn"};
char *p_names[]  = {"Paul", "Bob", "Emma", "Jim", "Kathryn"};
```

Which of the above is better?

Function Pointers

- It is possible to define pointers to functions
  - Functions are however **not** variables

```
int (*f_p)(int, int);
```

- `f_p` is a pointer to a function taking two integer arguments and returning an integer.
  - If `f` is a function then `&f` is a pointer to a function
  - Just in the same way that if `a` is an integer then `&a` is a pointer to an integer

```
int add(int a, int b);
int sub(int a, int b);

void main()
{
    int (*f_p)(int, int);
    f_p = &add;
    return;
}
```

Using function pointers

- Treat the function pointer like it is the function you want to call.
  - There is no need to dereference (`*f_p`) but you may if you wish

```
f_p = &add;
printf("add = %d\n", f_p(10, 4));
f_p = &sub;
printf("sub = %d\n", f_p(10, 4));
```

```
add = 14
sub = 6
```

- Care is needed with parenthesis
  - What is `f`?
  - What is `g`?

```
int *f();
int (*g)();
```

Using function pointers

- Treat the function pointer like it is the function you want to call.
  - There is no need to dereference (`*f_p`) but you may if you wish

```
f_p = &add;
printf("add = %d\n", f_p(10, 4));
f_p = &sub;
printf("sub = %d\n", f_p(10, 4));
```

```
add = 14
sub = 6
```

- Care is needed with parenthesis
  - What is `f`? function returning pointer to int
  - What is `g`? pointer to a function returning int

```
int *f();
int (*g)();
```

const pointers

- Remember the definition of `const`?
  - Not unintentionally modifiable

- What then is the meaning of the following?

```
char * const p;
```

```
const char * p;
```

```
char const * const p;
```

const pointers

- Remember the definition of `const`?
  - Not unintentionally modifiable
  - Read from right to left

https://cdecl.org/ - C Gibberish to English

- What then is the meaning of the following?

```
char * const p;
```

The pointer is constant but the data pointed to is not
i.e. declare p as const pointer to char

```
char const * p;
```

=

```
const char * p;
```

The pointed to data is constant but the pointer is not
i.e. declare p as pointer to const char

```
char const * const p;
```

The pointer is constant and the data it points to is also constant
i.e. declare p as const pointer to const char

Summary

- Advanced use of Pointers
  - Identify a general purpose pointer
  - Determine the endianness of a computing system
  - Interpret advanced pointer declarations
  - Recognise function pointers and determine where they may be used

- Next Lecture: Dynamically managed Memory