CSCI 4060U: OpenCL Programming Basics I

Host Types (C) vs Device Types (OpenCL C)

- Host and device may have different representations for data (e.g., twos complement)
- Host and device may also have different sized data (e.g., number of bytes)

Integers

```
typedef char            int8_t;
typedef unsigned char   uint8_t;
typedef short           int16_t;
typedef unsigned short  uint16_t;
typedef int             int32_t;
typedef unsigned int    uint32_t;
typedef long            int64_t;
typedef unsigned long   uint64_t;
```

Floating Points

```
double      //64 bits – might be supported
float       //32 bits – always be supported
half        //16 bits – might be supported
```

Device types (restricted)

```
bool  //Boolean type 1 or 0
size_t
ptrdiff_t
intptr_t
uintptr_t
```

Memory regions

- Every variable in a kernel needs to explicitly specify the memory region
- You always must think about location!

```
__global
__constant
__local
__private
```

Consider some examples!

Question #1: Is the below example legal in OpenCL C?

```
__global int* x;
__global int* y;
x = y;
```

Answer: YES!

Question #2: Is the below example also legal in OpenCL C?

```
__global int* x;
__private int* y;
x = y;
```

Answer: NO because x and y are in different types of memory

Question #3: Is this example legal in OpenCL?

```
__global int* x;
__private int* y;
*x = *y;
```

Answer: YES! You can move data from one type of memory to another by copying it.

OpenCL Vector Types

- Can be signed or unsigned (generally)

```
//Signed vector types
charN
shortN
intN
longN
floatN
doubleN

//Unsigned vector types
ucharN
ushortN
uintN
ulongN
```

where `N = {2,4,8,16}`

Example #1: Consider a vector-vector operation

```
int4 x, y, z;
…
z = x + y;
//(z1, z2, z3, z4) = (x1+y1, x2+y2, x3+y3, x4+y4)
```

Example #2: Consider a vector-scalar operation

```
int4 x;
int y;
int4 z;
…
z = x + (int4)y;
//(z1, z2, z3, z4) = (x1+y, x2+y, x3+y, x4+y)
```

Example #3: Accessing the individual elements/components of a vector

```
int4 x, y, z;
…
//component access is of form <vector_name>.<component>
//where component is s0,s1,…s9,SA,…,SF (use HEX 0-15)
z.s0 = x.s0 + y.s0;
//(z1) = (x1+y1)
```

Performance Considerations

Question: What is the benefit of OpenCL C vector types?

Answer: Performance is a big part of why we use OpenCL C vector types. Consider the following OpenCL C example again:

```
int4 x, y, z;
…
z = x + y;
```

It gets compiled into the following (pseudo) machine code:

```
vector_add_4xi16 r3, r1, r2
```