CSCI 4060U: Lecture 16 – OpenCL Programming II

Host data types (C) vs Device data type (OpenCL C)

- May have different representations (e.g. twos complements)
- May have different sizes (e.g. number of bytes)

Floating Points (on the device)

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

Integers (on the device)

```
• 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;
```

Device restricted types

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

Memory regions

- need to be specified for each variable in a kernel
- need to get used to *always* be thinking about location

- `__global`
- `__constant`
- `__local`
- `__private`

Question #1: Is the below OpenCL C code legal?

```
__global int* x;
__global int* y;
x = y; //YES! This is legal
```

Question #2 : Is the below OpenCL C code legal?

```
__private int* x;
__private int* y;
x = y; //YES! This is legal
```

Question #3: Is the below OpenCL C code legal?

```
__global int* x;
__private int* y;
x = y; //NO! Different types of memory
```

Question #4: How do we move data from one type of memory to another?

```
__global int* x;
__private int* y;
*x = *y; //YES! We copy the value not the pointer
```

OpenCL Vector Types

- Can be signed or unsigned

```
//SIGNED VECTOR TYPES
charN
shortN
intN
longN
floatN
doubleN

//UNSIGNED VECTOR TYPES
ucharN
ushortN
uintN
ulongN

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

Example #1: Vector-Vector Operations

```
int4 x, y, z;
..
z = x + y;
//{z1,z2,z3,z4} = {x1+y1,x2+y2,x3+y3,x4+y4}
```

Example #2: Vector-Scalar Operations

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

Example #3: Operations on Vector Components

```
int4 x, y, z;
..
z.s0 = x.s0 + y.s0;
//format is <vector_name>.<component>
//where component is s0, s1, s2, .. s9, sA, .. SF (Use HEX)
```

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

Answer: Performance! Consider the below OpenCL C code:

```
int4 x, y, z;
..
z = x + y;
```

It compiles to something like the following pseudocode:

```
vector_add_4x16 r3, r1, r2
```