Q1: What about integrated GPUs, sharing memory with CPU?
A: OpenACC will work with discrete memory and shared memory GPUs. In the case of shared memory GPUs data management directives are not necessary, but it's generally best to assume a discrete memory GPU and implement data directives so that your code will be able to run anywhere. On a shared memory machine OpenACC allows the compiler to ignore data directives that are unnecessary. Data management directives will be discussed in lecture 3 and its associated lab.

Q2: Does the RAM GPU memory refers to the GDDR memory?
A: Yes, on a CPU the RAM is typically DDR3 or DDR4 and on a GPU the RAM is referring to the GDDR5 memory on the board.

Q3: What does aliasing mean for pointer?
A: Aliasing occurs when two pointers point to the same memory. For instance, if I allocate two arrays, A and B, I know that these arrays will never overlap. If I pass those arrays into a function, however, the compiler will only see two pointers and will have no way of knowing that they do not overlap, so it must assume that they could overlap and avoid any optimizations that might cause wrong answers if the pointers do overlap. Imagine this case:

```
void doubleMe(double *in, double *out, int size)
{
 for (int i = 0; i < size; i++ )
 {
   out[i] = 2.0 * in[i];
 }
}
```

If I call the function with two different arrays (`doubleme(A,B,N)`) then the in and out pointers aren't aliased, because I've passed two separate arrays. If I call the function with the same array (`doubleMe(A,A,N)`), however, then the in and out arrays are aliased. *False aliasing* is a term for the case when the compiler has assumed that pointers might be aliased when they actually aren't. If I use the restrict keyword, I can promise the compiler that in and out will never alias, possibly enabling additional optimizations and parallelization.

```
void doubleMe(double *restrict in, double *restrict out, int size)
{
 for (int i = 0; i < size; i++ )
 {
   out[i] = 2.0 * in[i];
 }
}
```

Wikipedia has a great article about the restrict keyword and pointer aliasing at https://en.wikipedia.org/wiki/Restrict.

Q4: Does the nvvp CPU profiling work with OpenMP code?
A: Yes. You can control how the threaded timing is displayed with the following option.

```
--cpu-profiling-thread-mode <mode>
        Set the thread mode of CPU profiling. Allowed values:
            "separated" - Show separate profile for each thread
            "aggregated" - Aggregate data from all threads
```

Q5: Can the unified memory be larger than gpu memory?
A: On current NVIDIA GPUs the size of managed memory is restricted to fit within GPU memory. If you use the PGI -ta=tesla:managed option then all dynamically allocated memory used by the program must fit in GPU memory.

Q6: Does OpenACC work harmoniously with pinned memory? In CUDA I can program with pinned memory to make things much faster. Will OpenACC kill that?
A: Pinned memory provides faster PCIe transfers and enables asynchronous data transfers. OpenACC does work with pinned memory and the compiler may use pinned memory behind the scenes. The PGI compiler even has an option (-ta=tesla:pin) to force all dynamically allocated memory into pinned memory, which sometimes improves performance, but may limit how much total memory a program can use.

Q7: Which approach is safe: Kernel based or parallel loop based?
A: The kernels approach puts all of the responsibility of making sure a loop is safe to parallelize on the compiler. The parallel loop approach puts this responsibility on the programmer. This means that the kernels approach will always be safer if you're not confident about whether the code can be parallelized. It also means that there will be times when the compiler is overly-cautious, as was demonstrated in the lecture.

Q8: What does openACC do when a loop contains call to class functions
A: All function calls within OpenACC loops must either be inlined by the compiler or must be decorated with the acc routine directive. This directive will be discussed in lecture 4, or more information can be found in the OpenACC programming guide available in the OpenACC Toolkit or at http://bit.ly/openacc-guide.

Q9: Where exactly does unified memory exist? On cpu or gpu?
A: Unified Memory actually exists on both, depending on where it's needed. On most modern systems a pointer is a virtual address into memory that the operating system translates into a physical memory address. With Unified Memory, a single virtual address is used and the runtime will migrate the data to either a physical memory page on the CPU or a physical memory page on the GPU depending on where it's needed. So if I'm accessing from the CPU, the memory will be on the CPU and if I'm accessing from the GPU it will be on the GPU. It's important to note that you cannot access a unified memory pointer from the GPU and the CPU at the same time.

Q10: how are cpu cores and gpu cores different?
A: As Jeff discussed in the lecture. CPU cores are well optimized for serial instructions. GPU cores are good at parallel computing by launching numbers of threads.

Q12: Does the NVIDIA ACC kit work well on the Linux Mint Distro? Got win10, Debian and Mint... which one you recommend?
A: OTK only supports Linux systems for now. Make sure your system has glibc 2.3.4+ and CUDA-enabled GPU with Compute Capability 2.0 (cc20).

Q13: Following up on the AMD Q ... Does the OpenACC support work for the OSX operating system? Last time I checked, the OSX AMD GPU driver precluded the use of the PGI OpenACC.
A: PGI OpenACC compiler does support OSX with NVIDIA GPU, but not with AMD GPU. We also support Linux/Windows with AMD GPU.

Q14: how to find out what to use for -ta option if not a Tesla
A: pgaccelinfo. Then check the PGI Compiler Option.

Q15: Is there a linux command that will tell us which GPU is available?
A: pgaccelinfo command in our OTK. It will show all the available GPUs (NVIDIA or AMD).

Q16: What is the lowest version of cuda that's compatible with opeacc?
A: PGI OpenACC supports GPU cc20+. Please check https://developer.nvidia.com/cuda-gpus

Q17: are GPU units called cores too?
A: yes, GPU units are called cores too.

Q18: Does OpenACC also work with AMD GPU's?
A: Yes, AMD GPUs are also supported

Q19: how much of this is currently possible with other compilers (GNU, Intel, etc.)?
A: There several members of the OpenACC group including Cray and PGI. You can also use

Q20: If 2 or more GPUS are available, will OpenACC take advantage of them?
A: Yes, you can use multiple GPUs with OpenACC. We will cover it in the lecture on Nov 12th.