**Deep Bodra**

**5801 1841**

**DBI Week 1 (1/6 - 1/10)**

**Detailed explanation of the topics covered in class**

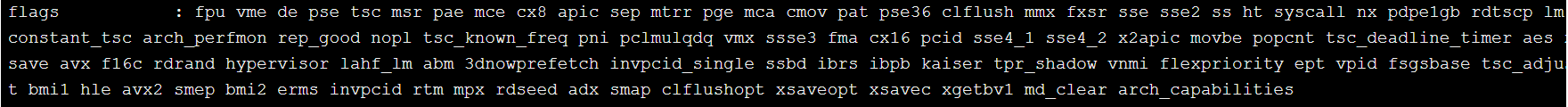
**1/7/2020**

1. Evolution of microprocessors
   1. 80386
      1. It was launched by Intel in 1985
      2. It uses 80387 for floating point calculations
      3. It has ACL unit to perform integer and logical operations
      4. The registers are connected to the ACL unit
   2. 80486
      1. It was launched by Intel in 1989
      2. It fabricated 80386 and 80487 on the same chip
      3. It borrowed techniques from expensive processors that were available at that time
      4. It incorporated the caching mechanism
      5. It made use of a large number of registers
      6. On an average the ADD and MUL instructions take 12-26 cycles
   3. Pentium 1995
      1. It made use of a super scalar architecture
      2. It has 3 ACL units
      3. It introduced **Pipelining**
         1. An instruction has to go through a lot of stages like Fetch, Decode, Execute.
         2. In pipelining these stages are connected one after the other like in an assembly line
         3. As a result, when the instruction ‘1’ moves to the next stage, another instruction ‘2’ is ready to be passed into the current stage. This increases the overall execution time
      4. To successfully implement pipelining it makes use of **Speculated Execution (which has branch prediction)** which runs through the code before it starts executing in order to fill the pipeline by predicting whether a branch will be taken or not
      5. The performance is highly dependent on the branch predictions. If the prediction turns out to be false then there is a penalty to be paid in which all the instructions should be emptied out from the pipeline. This is called **pipeline flushing**.
      6. To avoid pipeline flushing, we make use of **Out of Order execution** which loads instructions not necessarily in sequential order
   4. Pentium Pro 1998
      1. It is inspired by the concept of vectorization introduced by Cray Computing
      2. Vectorisation is also called as SIMD(Single Instruction Multiple Data) meaning the processor can perform an operation on multiple floating numbers in a clock cycle
      3. MMX instructions: can add 4 floats and 2 doubles
      4. SSE, SSE2,... SSE4: 4 comparisons in one clock cycle
      5. AVX, AVX2: can perform operations on 16 floats in one clock cycle

----------------------------------------------------------------------------------------------------------------------------------------

**Try at home**: Command to find instructions supported by the processor





----------------------------------------------------------------------------------------------------------------------------------------

**1/9/2020**

1. Code performance with respect to the computer architecture
   1. Performance counters helps to find:
      1. No of clock cycles spent
      2. No of branches
      3. No of branch miss predictions
      4. No of instructions executed
   2. For c++: Perf
2. Debugging in C++
   1. -o0: no optimizations at the compiler level
   2. -o3: code will run very fast when used arrays instead of using the fancy data structures
3. Memory Hierarchy
   1. CPU-Registers
      1. Memory access time is less than 1 clock cycle
   2. L1-cache
      1. It is on the same chip
      2. Memory access time is 1-2 clock cycles
   3. L2-cache
      1. It is on the motherboard
      2. Memory access time is 14-20 clock cycles
      3. Prefetching of instructions/data is done here
   4. DRAM
      1. Memory access time is 50-200 clock cycles
   5. Hard Disk
      1. Memory access time is 106 clock cycles
   6. Cache Line
      1. It is mostly 64B
      2. It makes use of:
         1. Temporal Locality: Data accessed at the moment must be cached as it may be required in the near future
         2. Spatial Locality: Data in the neighbouring regions of the data accessed right now must be cached as they may be required in the near future

--THANK YOU--