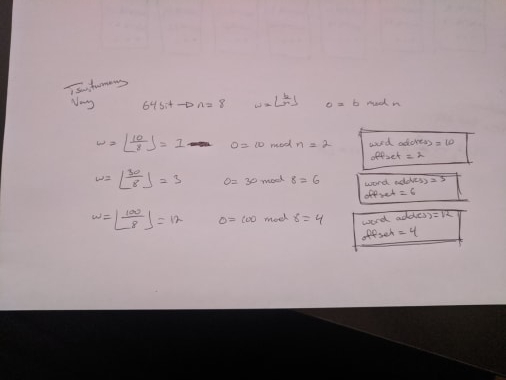
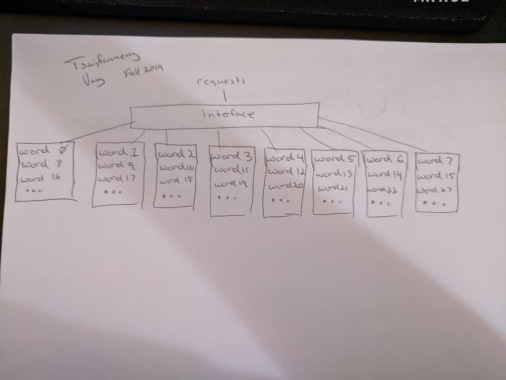
Tswjfwmeng Vang

Professor Ashan

Computer Architecture

December 16, 2019

Computer Architecture: Final

1. Assume a computer has a physical memory organized into 64-bit words. How many bits will be used as offset for word addressing? Give the word address and offset within the word for each of the following byte addresses: 10, 30 and 100. Justify your answer.
   1. **I think 64 bits would be used as offset for word addressing ?**
   2. 
2. If a bus can transfer 32 bits in each cycle and runs at a rate of 66 MHz, what is the bus throughput measured in megabytes per second? Show work.
   1. 66,000,000 \* 32 = 2,112,000,000 bits/s
   2. 2112000000 / 8 = 264,000,000 bytes/s
   3. 264,000,000 / 1000 = 264,000 Kb/s
   4. 264,000 / 1000 = **264 Mb/s**
3. Redraw Figure 11.13 from Comer’s book for an 8-way interleaved memory. When is using interleaved memory useful?
   1. 
   2. **It seems like using an interleaved memory would always be useful because the technique allows faster memory processing. According to Comer's book, if a text string is being searched, an interleaved memory could be used because it doesn’t need to wait for the memory to reset before moving on to the next module. This enables much faster searches because there is no wait time for the previous memory to reset.**
4. If a computer has 64-bit addresses and each address corresponds to one byte, how many gigabytes of memory can the computer address? (Show the math)
   1. 2 ^ 64 = 1.844674407 \* 10 ^ 19 bytes
   2. (1.844674407 \* 10 ^ 19) / 1000 = 1.844674407 \* 10^16 KB
   3. (1.844674407 \* 10^16) / 1000 = 1.844674407 \* 10 ^ 13 MB
   4. (1.844674407 \* 10 ^ 13) / 1000 = **1.844674407 \* 10 ^ 10 GB**
5. For a given piece of code, the hit ratio of first cache is 0.1 and the hit ratio of second cache is 0.3. The time required to access the first cache is 10 nanoseconds, the second cache is 100 nanoseconds, and the time to access the underlying physical memory is 1 microsecond, what is the effective memory access time for the piece of code? Show your work and justification.
   1. 0.1 (10 ns) + 0.9 (20 ns) = 19 ns
   2. 0.3 (100 ns) + 0.7 ( 200 ns) = 170 ns
   3. 19 ns + 170 ns + 0.001 ms = **189.001 ns**
6. Write an assembly program to store the cube values of first hundred numbers. Use bss to initialize an array of size 100 with 32-bit integers. Use iterative instructions to populate the array. Add comments
   1. **See folder for .asm file**
7. Write an assembly program to read two 8 byte integers with scanf and compute their greatest common divisor using Euclid's algorithm, which is based on the recursive definition
   1. **See folder for .asm file**