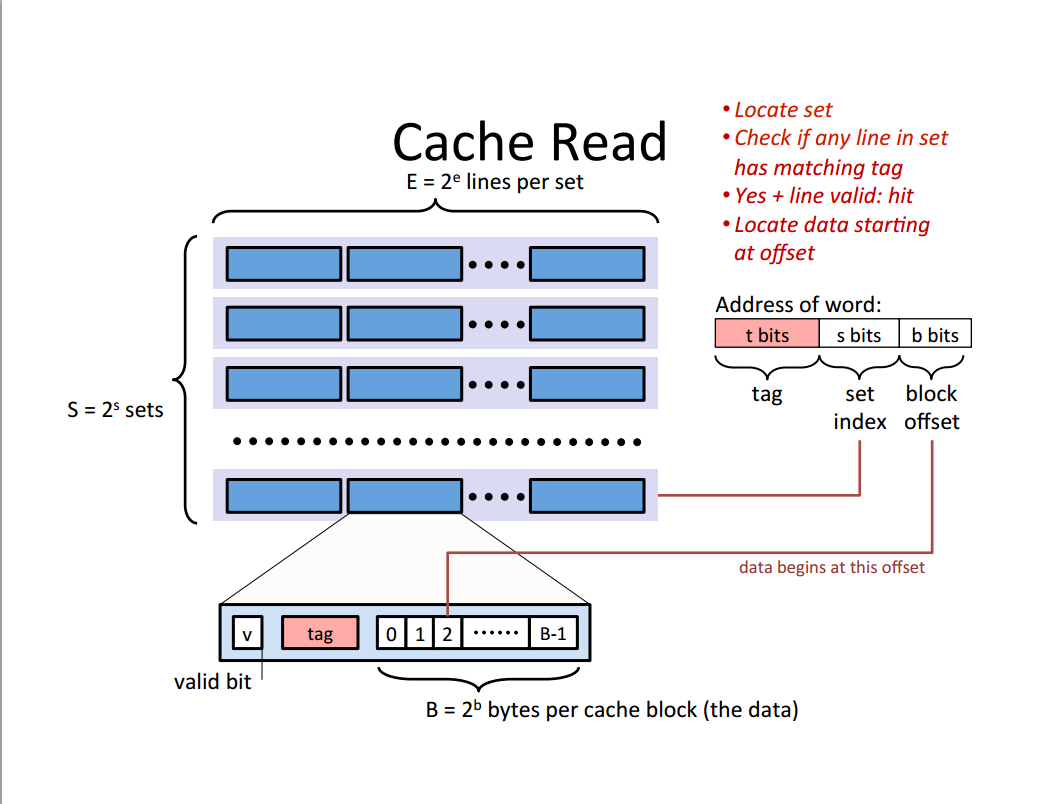
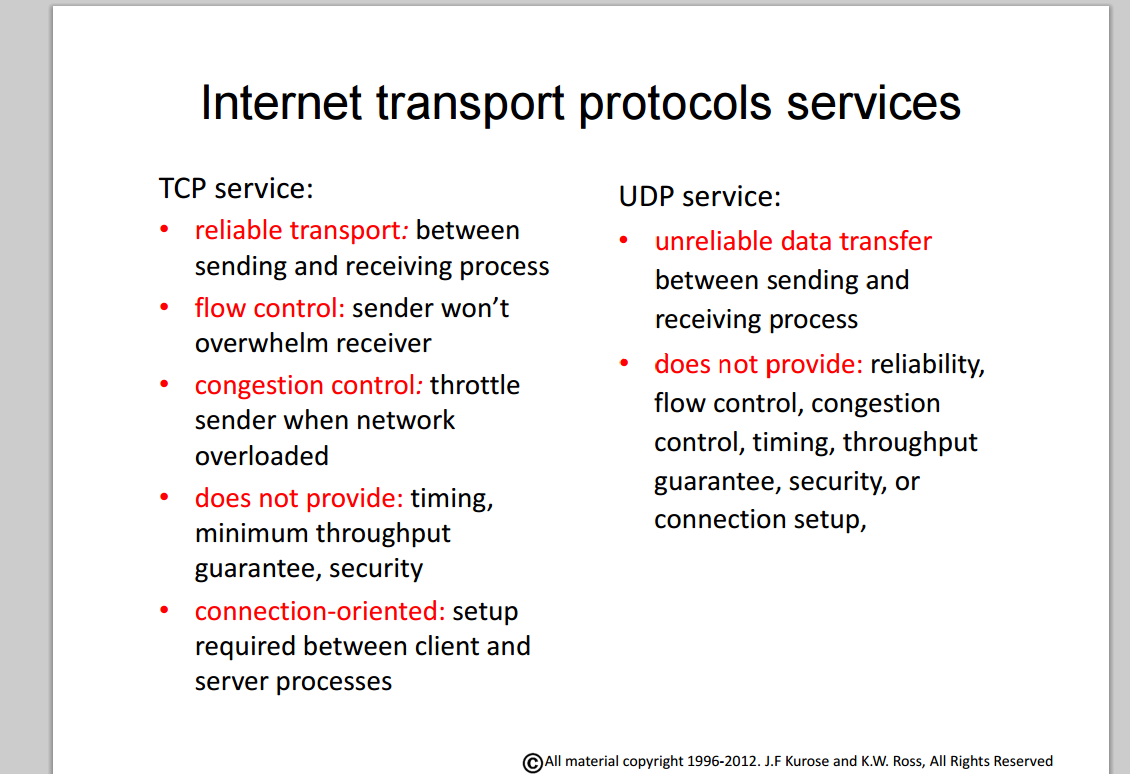
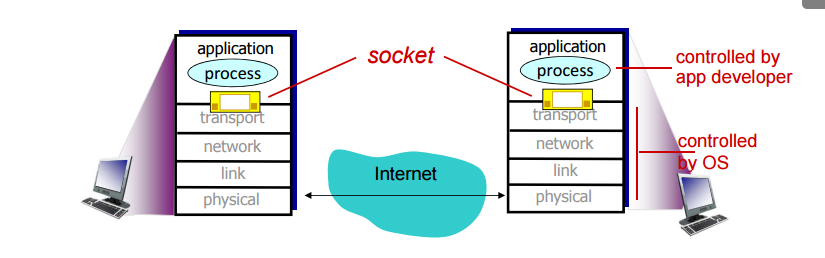
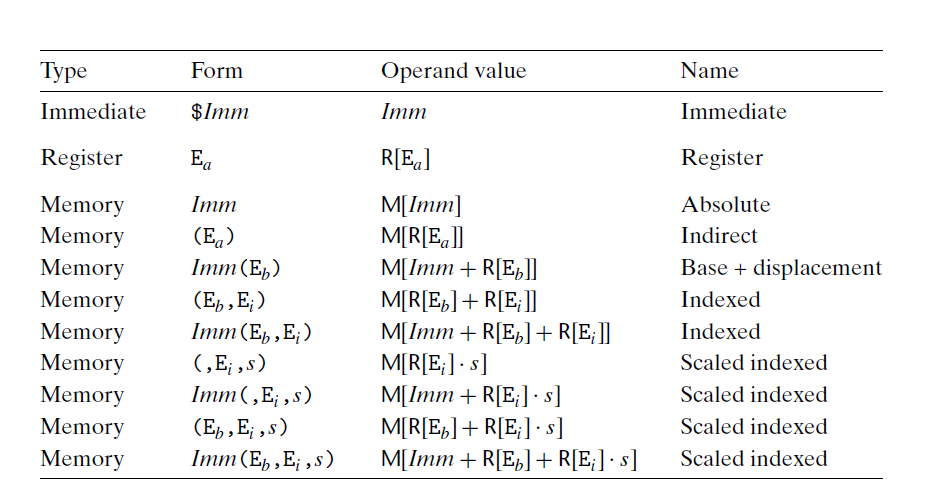
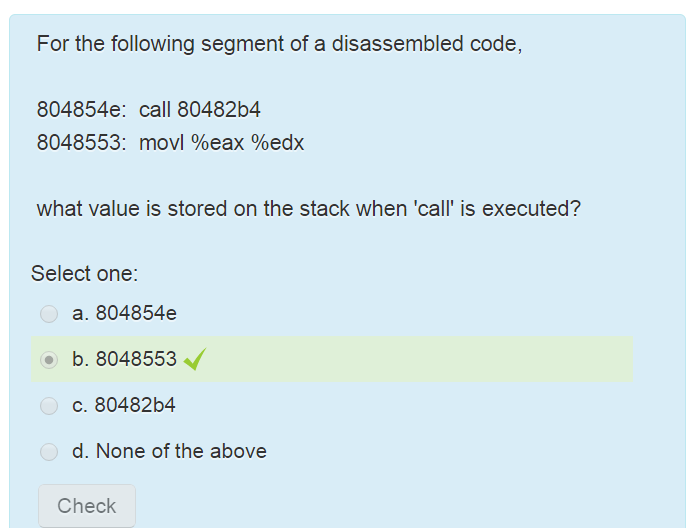
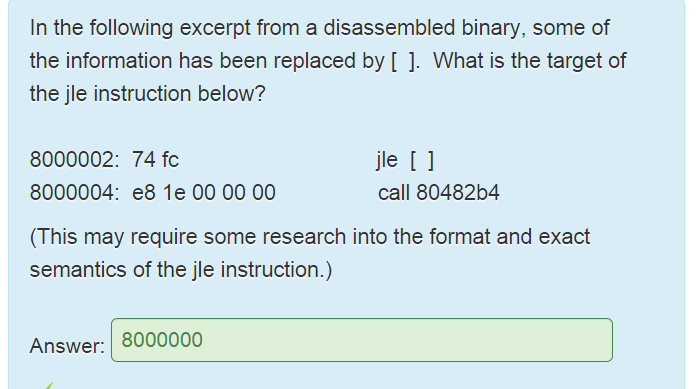
* 8(%ebp,%eax,4) … means: M[%ebp + (%eax\*4) + 8]
* leal ‐4(%eax,%ebx,2),%ecx lea = load effective address: calculates the address, but then uses that address as a value to store, in this case into %ecx Net effect: %ecx = %eax + (2\*%ebx) ‐ 4
* lea S,D load S into D move S,D loads value stored at address S into D
* LFU: Least Frequently Used: remove entry with LFU
* LRU: Least recently used: remove entry at the bottom
* Exploiting temporal locality - same data objects are likely to be reused multiple times(reuse in time)
* Exploiting spatial locality - block usually contain multiple data objects(use of nearby items)



* Direct mapped: one line per set (E=1) S=2^s sets
* L1 cache- smallest, 2 cycle latency
* L2- larger, 7 cycle
* L3, < 100cycle
* A write-hit occurs when we are writing a value that is currently stored in cache.
  + Write-Through- write immediately to memory
  + Write-Back – defer write to memory until replacement of line, need to determine if cache line is different from memory line.
* A write-miss occurs when we are writing a value that is no longer stored in cache.
  + Write allocate- fetch value from memory and percolate up cache hierarchy.
  + If no write allocate, write directly to memory.
* More about cache and miss rate including cycles: <http://www-edlab.cs.umass.edu/cs230/lecs/L18/cache-details.pdf>
* Block count=2^block bits
* Set count=2^set bits
* SIGSEGV-improper acces to a protected page, access to an unasked-for page
* Useful network byte-order conversion functions (I=32b, s=16b)
  + Htonl: convert uint32\_t from host to network byte order
  + Htons: convert uint16\_t from host to network byte order
  + Ntohl: convert 32\_t from network to host byte order
  + Ntohs: 16 from network to host byte order
* Functions for converting between binary IP address and dotted decimal strings:
  + Inet\_aton: dotted decimal string -> IP address in network byte order
  + Inet\_ntoa: IP address in network byte order ->dotted decimal string.
* 
* TCP Sockets-Ports
  + Well known- 0-1023
  + Registered: 1024-49151
  + Private/dynamic: 49152-65535
* Calls are blocking
  + Returns only after data is sent/received
* An EINVAL error occurs when the server is not listening for any incoming connections.
* The protocol can use SOCK\_STREAM is TCP.
* The transport layer aggregates data from different applications into a single stream before passing it to the network layer.
* 
* The correct order of function calls to establish connection in servers is bind, listen, accept.
* True about network protocol layering:
  + The layered architecture allows us to discuss a well-defined specific part of a large and complex system.
  + Makes it easier to change the implementation of the service provided by a layer.
  + Each network protocol belongs to one of the layers.
  + FALSE: Each layer does NOT use the service of layer below it.
* An endpoint of an inter-process communication flow across a computer network is network socket.
* The accept function call creates a connected file descriptor at the server that allows it to send and receive from a connecting client.
* Transmission control protocol (TCP)
  + Is a connection oriented protocol
  + Uses a three way handshake to establish a connection
  + Receives data from application as a single stream
* The memory management unit (MMU) translates virtual addresses to physical addresses.
* Virtual addresses of two different processes can map to the same physical address.
* Spatial locality: referencing array elements in succession.
* The size of the cache can be computed as SEB - > number of sets, number of lines, number of bytes per block.
* Virtual memory serves to
  + Give every application the illusion that it has the memory to itself.
  + Give applications the illusion that there’s more memory than there really is.
  + Protect applications from each other.
* The principle advantage of a write-through cache is lower complexity.
* The principle advantage of a write back cache is reduced memory traffic.
* Fully associative caches offer lower miss rates, while direct mapped caches have better latency.
* Virtual memory is organized as an array of contiguous byte-sized cells stored primarily in the disk
* About processes:
  + Each process share the CPU with other processes
  + Each process has its own private address space
  + The environment for a process is stored on the stack
  + FALSE: The operating system kernel runs as its own separate process
* Signals are not queued.
* SIGKILL cannot be blocked or ignored, SIGSTOP cannot be ignored.
* By default, SIGCHLD is ignored. Parents receive the child process id when it is stopped, interrupted, exit, or resumes after being interrupted.
* Once a blocked signal is unblocked, it will be handled by the process. A signal that is being ignored will never be handled.
* Calling exit() puts the process in terminated state.
* Process groups in Linux are useful because you can send a signal to all the processes in a process group at the same time.
* A child process that has finished execution and hasn’t been reaped is a zombie process.
* The arithmetic logic unit is the Execution Unit.
* Assembly and object code are NOT the same thing.
* Pushl = subl $4, %esp movl %ebp, (%esp)
* Popl = movl (%esp), %eax addl $4, %esp
* 
* Esp is a stack pointer
* 
* The control unit interprets instructions of the program and controls the other parts of the processor.
* To make a global variable private, declare it with static
* 
* IA32 does not allow direct transfer from one memory location to another, and no transfer of immediate data to memory. (ex. Movl (%eax), (%ebx, %ecx)
* Esp points to the top of the stack, stack grows toward lower addresses
* Characteristics of assembly operations:
  + Perform arithmetic function on register or memory data
  + Transfer data between memory and register
  + Transfer control
* Compile but not link –c
* Malloc -> heap
* Linking is performed after compile time
* Typecasting a structure is illegal
* Big Endian: store most significant byte in the smallest address
  + Little Endian: store least significant byte in the smallest address