

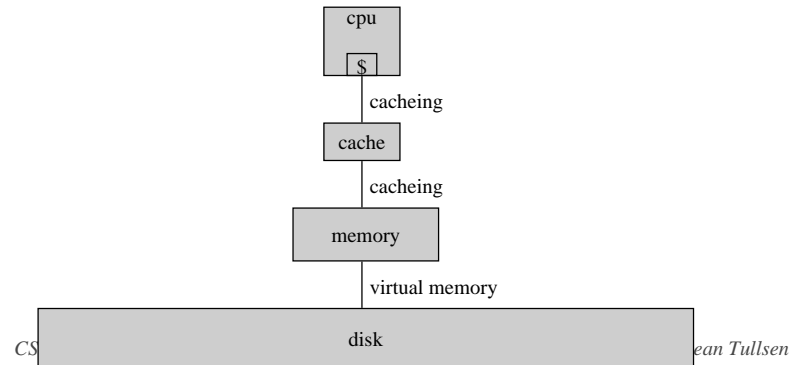
## Virtual Memory, Multiprocessing, Multithreading

CSE 240A

Dean Tullsen

## Virtual Memory

- It's just another level in the cache/memory hierarchy
- *Virtual memory* is the name of the technique that allows us to view main memory as a cache of a larger memory space (on disk).



## Virtual Memory

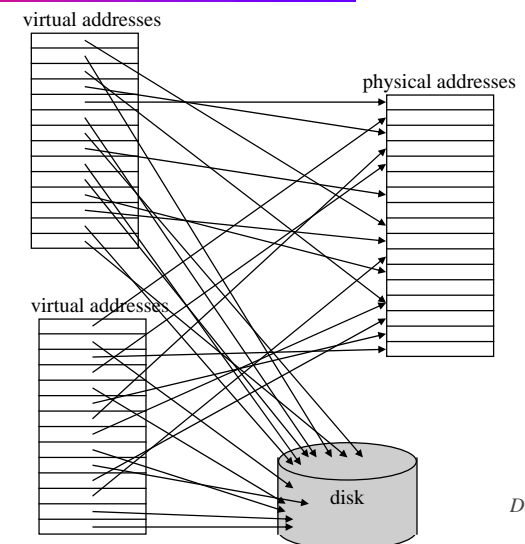
Provides

- performance (through the cacheing effect)
- protection
- ease of programming/compilation
- efficient use of memory

CSE 240A

Dean Tullsen

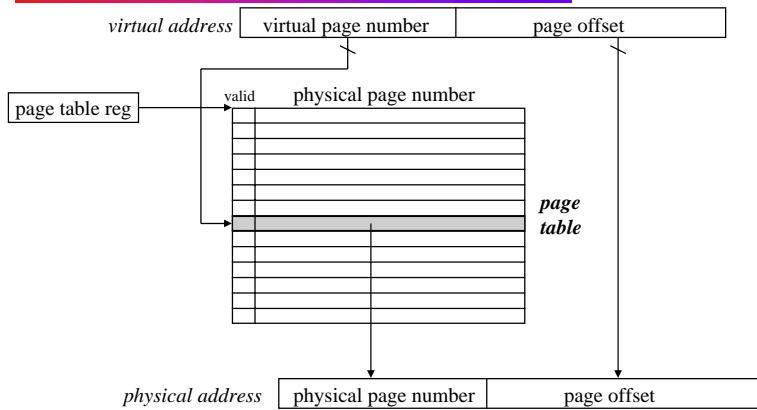
## Virtual Memory mapping



CSE 240A

Dean Tullsen

## Address translation via the page table



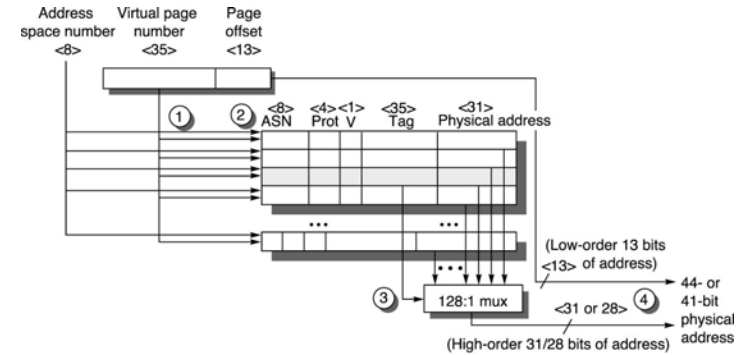
- all page mappings are in the page table, so hit/miss is determined solely by the valid bit (i.e., no tag)
- so why is this fully associative???

CSE 240A

Dean Tullsen

## Making Address Translation Fast

- A cache for address translations: translation lookaside buffer

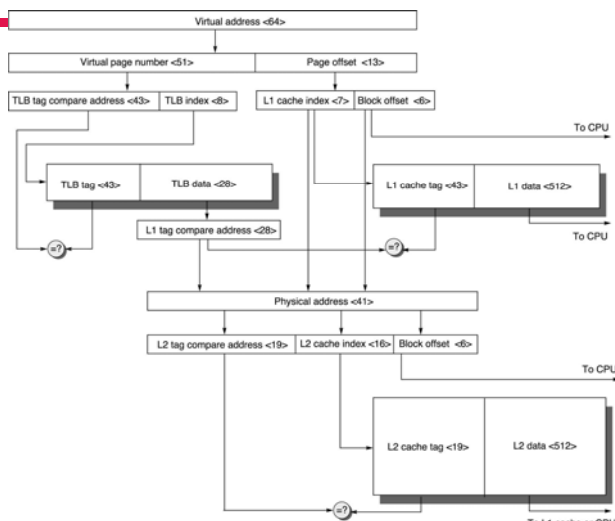


CSE 240A

ASN == address space number (basically, Process ID)

Dean Tullsen

## TLBs and caches



CSE 240A

Dean Tullsen

## Virtual Memory Key Points

- How does virtual memory provide:
  - protection?
  - sharing?
  - performance?
  - illusion of large main memory?
- Virtual Memory requires twice as many memory accesses, so we cache page table entries in the TLB.
- Three things can go wrong on a memory access: cache miss, TLB miss, page fault.

CSE 240A

Dean Tullsen

## Multiprocessors and Multiprocessing

*more is better?*

CSE 240A

Dean Tullsen

## Multiprocessors

- why would you want a multiprocessor?
- what things can it do well?
- What things can't it do well?
- What things can it do that a *bunch of computers* can't do?
- How much are you willing to pay?

CSE 240A

Dean Tullsen

## Classifying Multiprocessors

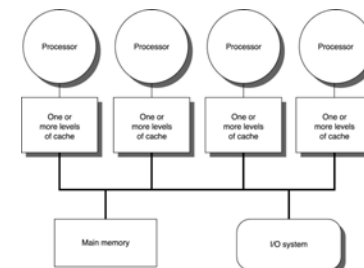
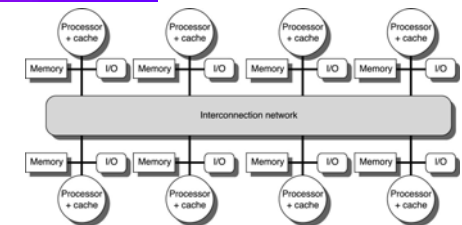
- Interconnection Network
- Memory Topology
- Programming Model

CSE 240A

Dean Tullsen

## Interconnection Network

- Bus
- Network
- pros/cons?

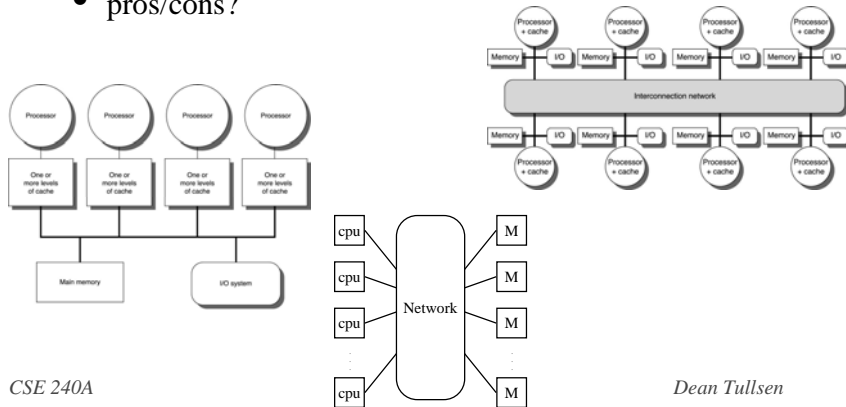


CSE 240A

Dean Tullsen

## Memory Topology

- UMA (Uniform Memory Access)
- NUMA (Non-uniform Memory Access)
- pros/cons?



CSE 240A

Dean Tullsen

## Programming Model

- Shared Memory -- every processor can name every address location
- Message Passing -- each processor can name only its local memory. Communication is through explicit messages (multicomputer).
- pros/cons?

- *find the max of 100,000 integers on 10 processors.*

CSE 240A

Dean Tullsen

## Parallel Programming

Processor A

$i = 47$

Processor B

index = i++;

index = i++;

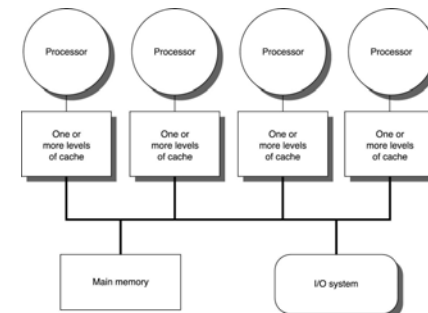
- Shared-memory programming requires synchronization to provide mutual exclusion and prevent race conditions
  - locks (semaphores)
  - barriers

CSE 240A

Dean Tullsen

## Multiprocessor Caches (Shared Memory)

- the problem -- cache coherency
- the solution?

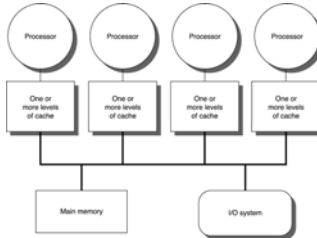


CSE 240A

Dean Tullsen

## Cache Coherency

- *write-update*
  - on each write, each cache holding that location updates its value
- *write-invalidate* <= most common
  - on each write, each cache holding that location invalidates the cache line.



- both schemes MUCH easier on a bus-based multiprocessor
- potentially requires a LOT of messages, but...

CSE 240A

Dean Tullsen

## Cache Coherency

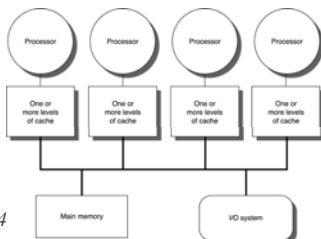
- A good cache coherency protocol can avoid sending unnecessary (and expensive) invalidate or update messages.
- Allows each cache line to be in one of several *states*.
- MESI (Illinois)
  - modified
  - exclusive
  - shared
  - invalid

CSE 240A

Dean Tullsen

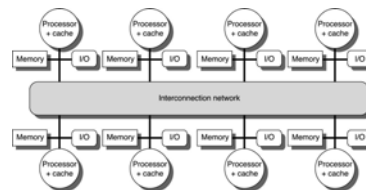
## Cache Coherency

- How do you know when an external read/write occurs?
- Snooping protocols
- Directory protocols



CSE 24

Dean Tullsen



## Multiprocessors – Key Points

- Network vs. Bus
- Message-passing vs. Shared Memory
- Shared Memory is more intuitive, but creates problems for both the programmer (memory consistency, requiring synchronization) and the architect (cache coherency).

CSE 240A

Dean Tullsen

## Simultaneous Multithreading

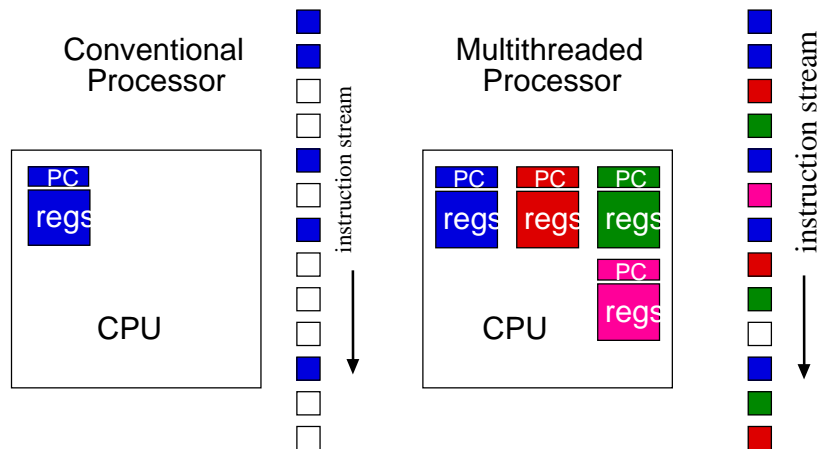
Tullsen, Eggers, Levy, *Simultaneous Multithreading: Maximizing On-Chip Parallelism*, ISCA, 1995

Tullsen, Eggers, Emer, Levy, Lo, Stamm, *Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor*, ISCA, 1996

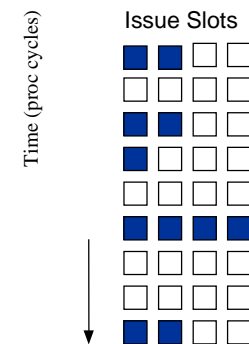
## Motivation

- Modern processors fail to utilize execution resources well.
- There is no single culprit.
- Attacking the problems one at a time (e.g., *specific* latency-tolerance solutions) always has limited effectiveness.
- However, a *general latency-tolerance solution* which can hide all sources of latency can have a large impact on performance.

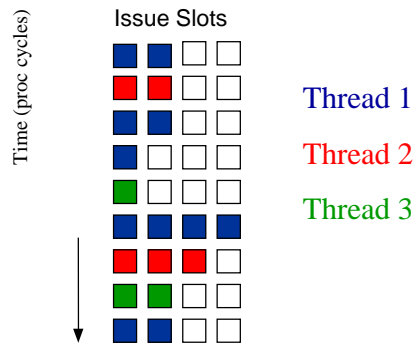
## Hardware Multithreading



## Superscalar Execution



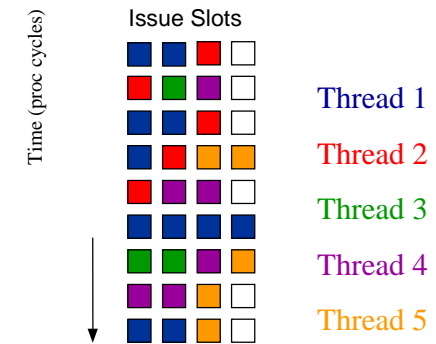
## Superscalar Execution with Fine-Grain Multithreading



CSE 240A

Dean Tullsen

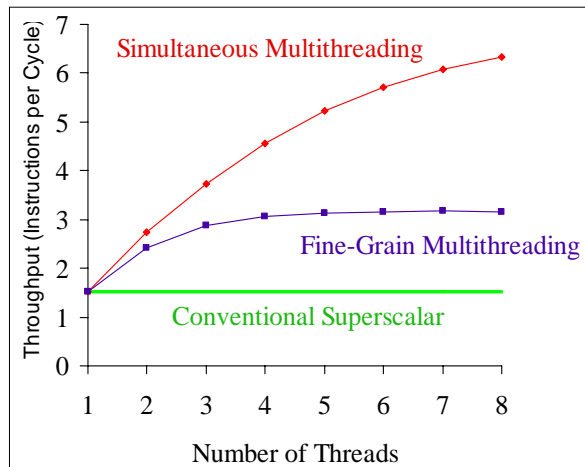
## Simultaneous Multithreading



CSE 240A

Dean Tullsen

## The Potential for SMT



CSE 240A

Dean Tullsen

## Goals

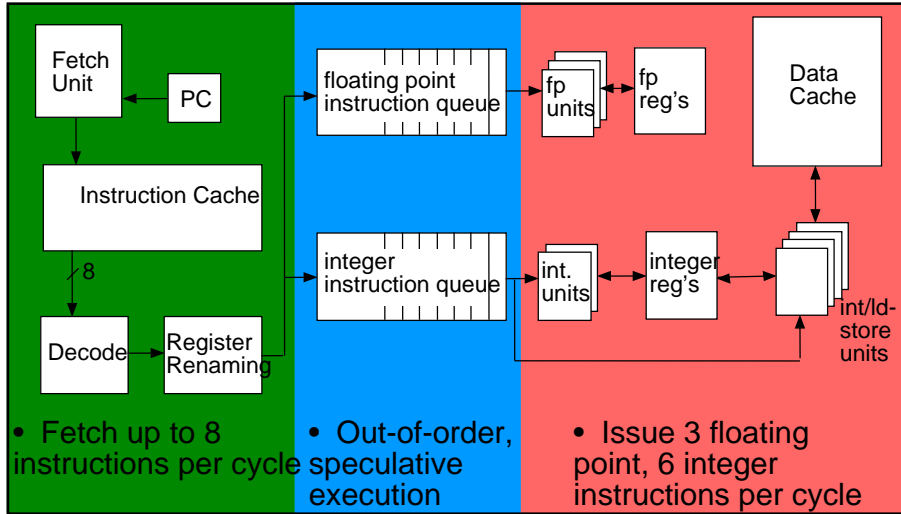
We had three primary goals for this architecture:

1. Minimize the architectural impact on conventional superscalar design.
2. Minimize the performance impact on a single thread.
3. Achieve significant throughput gains with many threads.

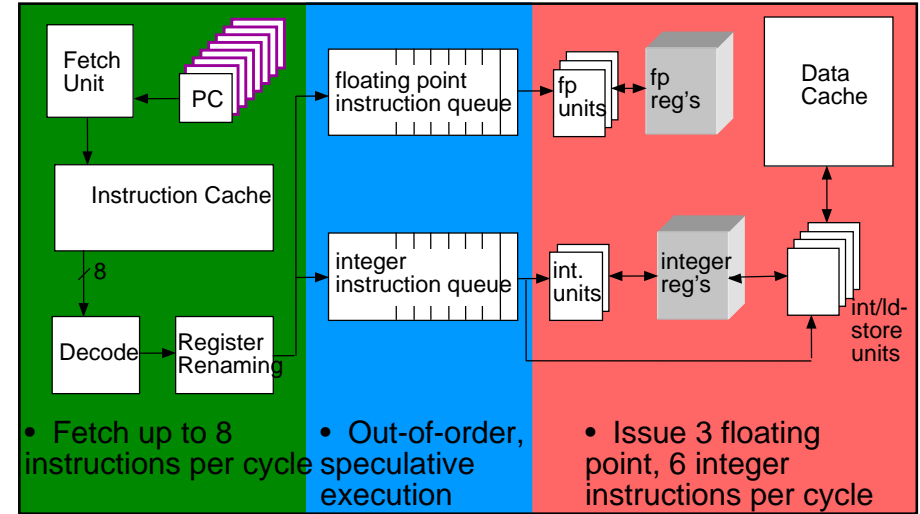
CSE 240A

Dean Tullsen

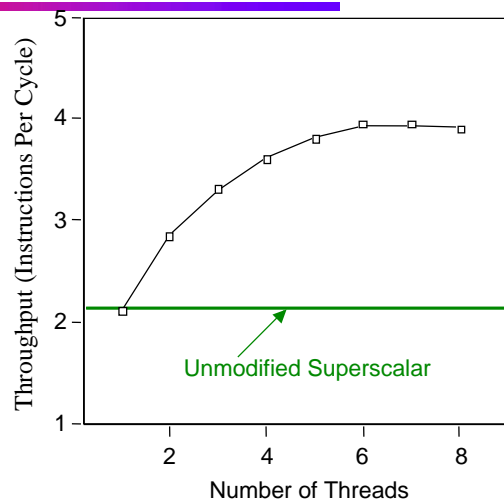
## A Conventional Superscalar Architecture



## An SMT Architecture



## Performance of the Naïve Design



## Bottlenecks of the Baseline Architecture

- Instruction queue full conditions (12-21% of cycles)
  - Lack of parallelism in the queue.
- Fetch throughput (4.2 instructions per cycle when queue not full)

## Improving Fetch Throughput

- The fetch unit in an SMT architecture has two distinct advantages over a conventional architecture.
  1. Can fetch from **multiple threads** at once.
  2. Can **choose which threads** to fetch.

CSE 240A

Dean Tullsen

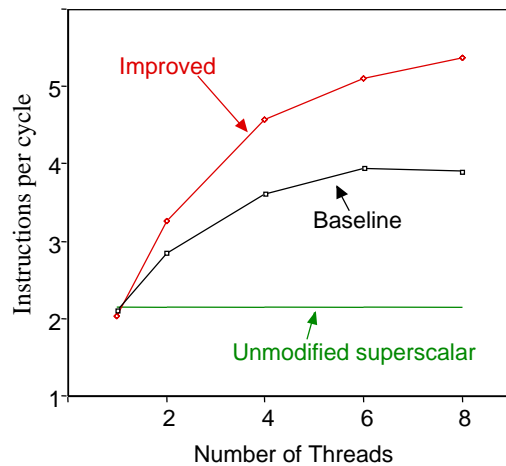
## Improved Fetch Performance

- Fetching from 2 threads/cycle achieved most of the performance from multiple-thread fetch.
- Fetching from the thread(s) which have the fewest unissued instructions in-flight significantly increases parallelism and throughput.

CSE 240A

Dean Tullsen

## Improved Performance



CSE 240A

Dean Tullsen

## This SMT Architecture, then:

- Borrows heavily from conventional superscalar design.
- Minimizes the impact on single-thread performance
- Achieves significant throughput gains over the superscalar (2.5X, up to 5.4 IPC).

CSE 240A

Dean Tullsen

## Commercial SMT

---

- Alpha 21464 (⊗)
- Clearwater Networks CNP810SP Network Services Processor
- Intel Pentium 4 “hyper-threading” processor.
- IBM Power 5 – 2 cores, 2 SMT threads/core
- Sun Niagara (2006) – 8 cores, 4 threads/core (SMT?)