Review

- Implementing precise interrupts in in-order pipelines:
  - Save exceptions in pipeline until commit point
  - Check for traps and interrupts before commit
- No architectural state overwritten before commit
- Support multithreading with translation and protection
  - Base and bound, simple scheme, suffers from memory fragmentation
  - Paged systems remove external fragmentation but add indirection through page table

A Problem in the Early Sixties

- There were many applications whose data could not fit in the main memory, e.g., payroll
  - Paged memory system reduced fragmentation but still required the whole program to be resident in the main memory

Manual Overlays

- Assume an instruction can address all the storage on the drum
- **Method 1:** programmer keeps track of addresses in the main memory and initiates an I/O transfer when required
  - Difficult, error prone!
- **Method 2:** automatic initiation of I/O transfers by software address translation
  - Brooker’s interpretive coding, 1960
  - Inefficient!

Not just an ancient black art, e.g., IBM Cell microprocessor used in Playstation-3 has explicitly managed local store!
Demand Paging in Atlas (1962)

"A page from secondary storage is brought into the primary storage whenever it is (implicitly) demanded by the processor."
Tom Kilburn

Primary memory as a cache for secondary memory
User sees 32 x 6 x 512 words of storage

Hardware Organization of Atlas

Initial Address Decode
16 ROM pages 0.4 – 1 µsec
2 subsidiary pages 1.4 µsec
Main 32 pages 1.4 µsec
Drum (4) 192 pages

Compare the effective page address against all 32 PARs
match → normal access
no match → page fault
save the state of the partially executed instruction

Recap: Typical Memory Hierarchy

Take advantage of the principle of locality to present the user with as much memory as is available in the cheapest technology at the speed offered by the fastest technology

Modern Virtual Memory Systems

Illusion of a large, private, uniform store

Protection & Privacy
several users, each with their private address space and one or more shared address spaces
page table = name space

Demand Paging
Provides the ability to run programs larger than the primary memory
Hides differences in machine configurations
The price is address translation on each memory reference

Administrivia

• Regrade request deadline Monday Nov 26
  – For everything up to Project 4
CS61C in the News
“World’s oldest digital computer successfully reboots”
Iain Thomson
The Register, 11/20/2012

Aker three years of restoration by the National Museum of Computing (TNMOC) and staff at Bletchley Park, the world’s oldest functioning digital computer has been successfully rebooted at a ceremony attended by two of its original developers. The 2.5-ton Harwell Dekatron, later renamed the Wolverhampton Instrument for Teaching Computation from Harwell (WITCH), was first constructed in 1949 and from 1951 ran at the USA’s Harwell Atomic Energy Research Establishment, where it was used to process mathematical calculations for Britain’s nuclear program.

The system uses 828 flashing Dekatron valves, each capable of holding a single digit, for volatile memory, plus 480 GPO 3000 type relays to shift calculations and 64 paper tape readers. It was very slow, taking a couple of seconds for each addition or subtraction, five seconds for multiplication and up to 15 for division.

Hierarchical Page Table

Address Translation & Protection

Translation Lookaside Buffers (TLB)

TLB Designs

Two-Level Page Tables in Physical Memory

Two-Level Page Tables in Physical Memory
Handling a TLB Miss

Software (MIPS, Alpha)
TLB miss causes an exception and the operating system walks the page tables and reloads TLB. A privileged "untranslated" addressing mode used for walk

Hardware (SPARC v8, x86, PowerPC, RISC-V)
A memory management unit (MMU) walks the page tables and reloads the TLB.

If a missing (data or PT) page is encountered during the TLB reloading, MMU gives up and signals a Page-Fault exception for the original instruction.

Flashcard Quiz:
Which statement is false?

- TLB miss is much faster than page fault
- TLB exploits spatial locality
- TLB hardware grows with larger page size
- TLB exploits temporal locality

Hierarchical Page Table Walk:
SPARC v8

Virtual Address

Context Table
Register

Context Table
Register

Physical Address

MMU does this table walk in hardware on a TLB miss

Page-Based Virtual-Memory Machine
(Hardware Page-Table Walk)

- Assumes page tables held in untranslated physical memory

Address Translation:
putting it all together

Virtual Address

TLB Lookup

Page Table Walk

Protection Check

Page Fault (OS loads page)

Protection Fault

Physical Address (to cache)

TLB miss? Inst. Cache

Page Fault?

Inst. TLB

TLB miss? Data TLB

Page Fault?

Data Cache

TLB miss? Protection violation?

TLB miss? Protection violation?

Handling VM-related traps

- Handling a TLB miss needs a hardware or software mechanism to refill TLB
- Handling a page fault (e.g., page is on disk) needs a restartable trap so software handler can resume after retrieving page
  - Precise exceptions are easy to restart
  - Can be imprecise but restartable, but this complicates OS software
- Handling protection violation may abort process
**Address Translation in CPU Pipeline**

- TLB miss? Page Fault? Protection violation?

  • Need to cope with additional latency of TLB:
    - slow down the clock?
    - pipeline the TLB and cache access?
    - virtual address caches (see CS152)
    - parallel TLB/cache access

**Concurrent Access to TLB & Cache (Virtual Index/Physical Tag)**

- Index L is available without consulting the TLB
  → cache and TLB accesses can begin simultaneously!
  Tag comparison is made after both accesses are completed

**Virtual-Index Physical-Tag Caches:**

- After the PPN is known, 2^b physical tags are compared

**VM features track historical uses:**

- Bare machine, only physical addresses
  - One program owns entire machine
- Batch-style multiprogramming
  - Several programs sharing CPU while waiting for I/O
  - Base & bound: translation and protection between programs (not virtual memory)
  - Problem with external fragmentation (holes in memory), needed occasional memory defragmentation as new jobs arrived
- Time sharing
  - More interactive programs, waiting for user. Also, more jobs/second.
  - Motivated move to fixed-size page translation and protection, no external fragmentation (but now internal fragmentation, wasted bytes in page)
  - Motivated adoption of virtual memory to allow more jobs to share limited physical memory resources while holding working set in memory
- Virtual Machine Monitors
  - Run multiple operating systems on one machine
  - Idea from 1970s IBM mainframes, now common on laptops
    - e.g., run Windows on top of Mac OS x
  - Hardware support for two levels of translation/protection
    - Guest OS virtual -> Guest OS physical -> Host machine physical
  - Also basis of Cloud Computing
    - Virtual machine instances for Project 1

**Acknowledgements**

- These slides contain material developed and copyright by:
  - Arvind (MIT)
  - Krste Asanovic (MIT/UCB)
  - Joel Emer (Intel/MIT)
  - James Hoe (CMU)
  - John Kubiatowicz (UCB)
  - David Patterson (UCB)

- MIT material derived from course 6.823
- UCB material derived from course CS252