

Virtual Memory (VM) Ch 7.3

Memory Management
Address Translation
Paging
Hardware Support
VM and Cache

19/09/2001

Copyright Teemu Kerola 2001

1

Other Problems Often Solved with VM ⁽³⁾

- If you must want to have many processes in memory at the same time, how do you keep track of memory usage?
- How do you prevent one process from touching another process' memory areas?
- What if a process needs more memory than we have?

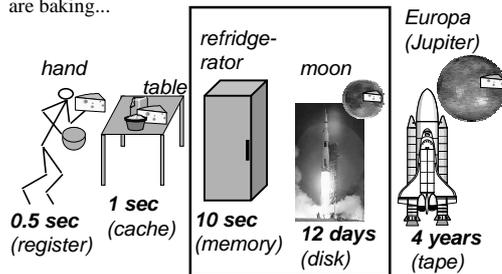
19/09/2001

Copyright Teemu Kerola 2001

4

Teemu's Cheesecake

Register, on-chip cache, memory, disk, and tape speeds relative to times locating cheese for the cheese cake you are baking...



19/09/2001

Copyright Teemu Kerola 2001

2

Memory Management Problem ⁽⁴⁾

- How much memory for each process?
 - is it fixed amount during the process run time
 - or can it vary during the run time?
- Where should that memory be?
 - in a continuous or discontinuous area?
 - is the location the same during the run time
 - or can it vary dynamically during the run time?
- How is that memory managed?
- How is that memory referenced?

19/09/2001

Copyright Teemu Kerola 2001

5

Virtual Memory

(virtuaalimuisti)

Ch 7.3

- Problem: How can I make my (main) memory as big as my disk drive?
- Answer: Virtual memory
 - keep only most probably referenced data in memory, and rest of it in disk
 - disk is much bigger and slower than memory
 - address in machine instruction may be different than memory address
 - need to have efficient address mapping
 - most of references are for data in memory
 - joint solution with HW & SW

19/09/2001

Copyright Teemu Kerola 2001

3

Partitioning ⁽³⁾

- How much physical memory for each process?
- Static (fixed) partitioning (staattiset tai kiinteät partitiot)
 - amount of physical memory determined at process creation time
 - continuous memory allocation for partition
- Dynamic partitioning (dynaamiset partitiot)
 - amount of physical memory given to a process varies in time
 - due to process requirements (of this process)
 - due to system (I.e., other processes) requirements

19/09/2001

Copyright Teemu Kerola 2001

6

Static Partitioning

- Equal size - give everybody the same amount
 - fixed size - big enough for everybody
 - too much for most Fig. 7.14
 - need more? Can not run!
 - internal fragmentation (sisäinen pirstoutuminen)
- Unequal size
 - external fragmentation (ulkoinen pirstoutuminen)
- Variable size Fig. 7.15
 - external fragmentation

19/09/2001 Copyright Teemu Kerola 2001 7

Address Mapping

logical address

Textual machine language:
1312: LOAD R1, 2510

+100000?

Execution time:
101312: LOAD R1, 102510 or
101312: LOAD R1, 2510 ??

physical address (constant?)

- Want: R1 ← Mem[102510] or Mem[2510] ?
- Who makes the mapping? When?

19/09/2001 Copyright Teemu Kerola 2001 10

Dynamic Partitioning (3)

- Process must be able to run with different amounts of main memory
 - all of memory space is **not** in physical memory
- New process?
 - reduce amount of memory for some (lower priority) processes
- Not enough memory for some process?
 - reduce amount of memory for some (lower priority) processes
 - kick (swap) out some (lower priority) process

19/09/2001 Copyright Teemu Kerola 2001 8

Address Mapping (2)

- At program load time
 - loader (lataaja)
 - static address binding (staattinen osoitteiden sidonta)
- At program execution time
 - cpu
 - with every instruction
 - dynamic address binding (dynaaminen osoitteiden sidonta)
 - swapping
 - virtual memory

19/09/2001 Copyright Teemu Kerola 2001 11

Address Mapping (4) (osoitteen muunnos)

Pascal, Java:
while (...)
X := Y+Z;

Symbolic Assembler:
loop: LOAD R1, Y
ADD R1, Z
STORE R1, X

Textual machine language:
1312: LOAD R1, 2510
ADD R1, 2514
STORE R1, 2600
(addresses relative to 0)

Execution time:
101312: LOAD R1, 102510
ADD R1, 102514
ADD R1, 102600
(real, actual!)

19/09/2001 Copyright Teemu Kerola 2001 9

Swapping (4) (heittovaihto)

- Keep all memory areas for all running and ready-to-run processes in memory
- New process
 - find continuous memory partition and swap the process in
- Not enough memory?
 - Swap some (lower priority) process out
- Some times can swap in only (runnable) portions of one process
- Address map: add base address

19/09/2001 Copyright Teemu Kerola 2001 12

VM Implementation (2)

- Methods
 - base and limit registers
 - segmentation
 - paging
 - segmented paging
- Hardware support
 - MMU - Memory Management Unit
 - part of processor
 - varies with different methods
 - Sets limits on what types of virtual memory (methods) can be implemented using this HW

19/09/2001

Copyright Teemu Kerola 2001

13

Segmentation Address Mapping

- Segment table
 - maps segment id to physical segment base address and to segment size
- Physical address:
 - find entry in segment table
 - check: byte offset < segment size
 - physical address: base + byte offset

19/09/2001

Copyright Teemu Kerola 2001

16

Base and Limit Registers (2)

- Continuous memory partitions
 - one or more (4?) per process
 - may have separate base and limit registers
 - code, data, shared data, etc
 - by default, or given explicitly in each mem. ref.
- *BASE* and *LIMIT* registers in MMU
 - all addresses logical in machine instructions
 - address mapping for address (x):
 - check: $x < LIMIT$
 - physical address: $BASE+x$

19/09/2001

Copyright Teemu Kerola 2001

14

Paging

- Process address space divided into (relatively small) equal size pages
 - address space division is not based on logical entities, only on fixed size chunks
- Each page is allocated its own physical page frame in memory
 - any page frame will do!
- Internal fragmentation
- Memory addresses have two fields

01100110 10110000
 page byte offset (lisäys)

19/09/2001

Copyright Teemu Kerola 2001

17

Segmentation (5)

- Process address space divided into (relatively large) logical segments
 - code, data, shared data, large table, etc
- Each logical segment is allocated its own continuous physical memory segment
- External fragmentation
- Memory address have two fields

011001 1010110000
 segment byte offset (lisäys)

19/09/2001

Copyright Teemu Kerola 2001

15

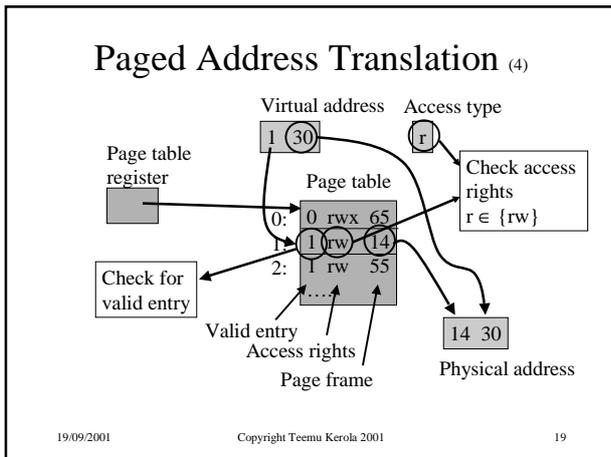
Paged Address Mapping

- Page table
 - maps page nr to physical page frame
- Physical address:
 - find entry in page table
 - physical address: page address + byte offset

19/09/2001

Copyright Teemu Kerola 2001

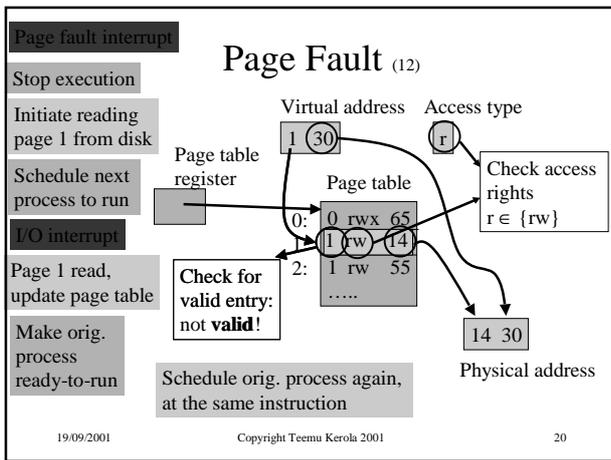
18



Address Translation (3)

- MMU does it for every memory access
 - code, data
 - more than once per machine instruction!
- Can not access page tables in memory every time - it would be too slow!
 - too high cost to pay for virtual memory?
- MMU has a “cache” of most recent address translations
 - TLB - Translation Lookaside Buffer (osoiteenmuunnos-taulukko)
 - 99.9% hit ratio?

19/09/2001 Copyright Teemu Kerola 2001 22



Translation Lookaside Buffer (3)

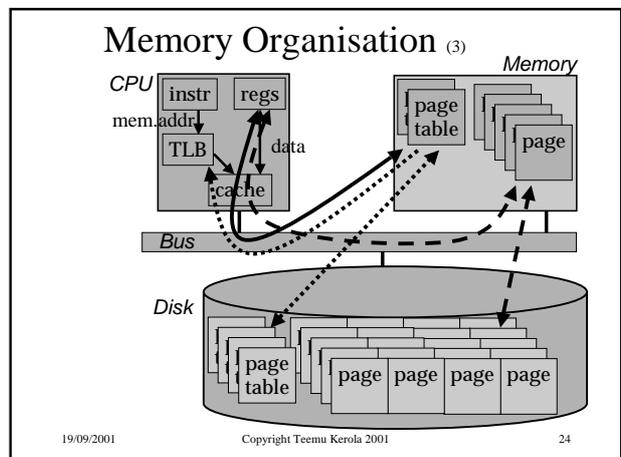
- “Hit” on TLB? (Fig. 7.19)
 - address translation is in TLB - real fast
- “Miss” on TLB?
 - must read page table from memory
 - takes time
 - cpu waits idle until it is done
- Just like normal cache, but for address mapping
 - implemented just like cache
 - instead of cache line data have physical address

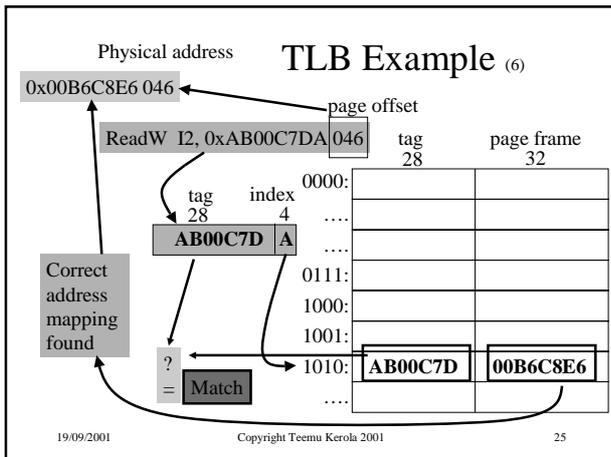
19/09/2001 Copyright Teemu Kerola 2001 23

Paging

- Physical memory partitioning
 - discontinuous areas (Fig. 7.16)
- Page tables
 - each process has its own
 - located in memory
 - can be very big
 - entry for each page in address space
- Inverted page table
 - entry for each page in memory (Fig. 7.18)
 - less space, more complex hashed lookup

19/09/2001 Copyright Teemu Kerola 2001 21





TLB Misses vs. Page Faults

TLB Miss

- CPU waits idling
- HW implementation
- Data is copied from memory to TLB
- Delay 4 (?) clock cycles

Page Fault

- Process is suspended and cpu executes some other process
- SW implementation
- Data is copied from disk to memory
- Delay 30 ms (?)

19/09/2001 Copyright Teemu Kerola 2001 28

- ### TLB and Cache (3)
- Usually address translation first and then cache lookup Fig. 7.20
 - Cache can be based on virtual addresses
 - can do TLB and cache lookup simultaneously
 - faster
 - Implementations are very similar
 - TLB often fully associative
 - optimised for temporal locality (of course!)
- 19/09/2001 Copyright Teemu Kerola 2001 26

- ### Virtual Memory Policies (3)
- Fetch policy (noutopolitiikka)
 - demand paging: fetch page only when needed 1st time
 - working set: keep all needed pages in memory
 - prefetch: guess and start fetch early
 - Placement policy (sijoituspolitiikka)
 - any frame for paged VM
 - Replacement policy (poistopolitiikka)
 - local, consider pages just for this process for replacement
 - global, consider also pages for all other processes
 - dirty pages must be written to disk (likaiset, muutetut)
- 19/09/2001 Copyright Teemu Kerola 2001 29

TLB vs. Cache

TLB Miss

- CPU waits idling
- HW implementation
- Invisible to process
- Data is copied from memory to TLB
 - from page table data
- Delay 4 (or 2 or 8?) clock cycles

Cache Miss

- CPU waits idling
- HW implementation
- Invisible to process
- Data is copied from memory to cache
 - from page data
- Delay 4 (or 2 or 8?) clock cycles

19/09/2001 Copyright Teemu Kerola 2001 27

- ### Page Replacement Policy (2)
- Implemented in SW
 - HW support
 - extra bits in each page frame
 - M = Modified
 - R = Referenced
 - set (to 1) with each reference to frame
 - reset (to 0) every now and then
 - special (privileged) instruction from OS
 - automatically (E.g., every 10 ms)
 - Other counters?
- 19/09/2001 Copyright Teemu Kerola 2001 30

Page Replacement Policies ⁽⁶⁾

(sivunpoisto-
algoritmit)

OS
Virtual Memory
Management

- OPT - optimal
- NRU - not recently used
- FIFO - first in first out
 - 2nd chance
 - clock
- Random
- LRU - least recently used
 - complex counter needed
- NFU - not frequently used

19/09/2001 Copyright Teemu Kerola 2001 31

Page Fault Frequency (PFF) Dynamic Memory Allocation

- Two bounds: L=Lower and U=Upper
- Physical memory split into fixed size pages
- At every page fault
 - T=Time since previous page fault
 - if $T < L$ then give more memory
 - 1 page frame? 4 page frames?
 - if $U < T$ then take some memory away
 - 1 page frame?
 - if $L < T < U$ then keep current allocation

19/09/2001 Copyright Teemu Kerola 2001 34

Thrashing

- Too high mpl
- Too few page frames per process
 - E.g., only 1000? 2000?
 - Less than its working set
- Once a process is scheduled, it will very soon reference a page not in memory
 - page fault
 - process switch

19/09/2001 Copyright Teemu Kerola 2001 32

VM Summary ⁽⁵⁾

- How to partition memory?
 - Static or dynamic size (amount)
- How to allocate memory
 - Static or dynamic location
- Address mapping
- HW help (TLB) for address translation
 - before or concurrently with cache access?
- VM policies
 - fetch, placement, replacement

19/09/2001 Copyright Teemu Kerola 2001 35

