WE16.1

Bookmark this page

Video explanation of solution is provided below the problem.

Virtual Memory

10 points possible (ungraded)

Apple's iWatch is rumored to include an embedded Beta processor, referred to as the "micro Beta". The micro Beta design includes an MMU (Memory Management Unit) that supports 256 (2^8) bytes per page but only implements a 16-bit virtual address, i.e., the MMU ignores address bits [31:16] coming from the CPU. The MMU and the page fault handler implement an LRU replacement strategy. The design also includes 1M (2^{20}) bytes of physical memory and a large Flash memory that serves as a "disk."

1. Apple's engineers are thinking about implementing the page map using a separate SRAM memory with L entries, where each entry has B bits. If the page map includes the standard dirty and resident bits, what are the appropriate values for the parameters L and B? You can provide your answer in the form "2^N" to express a power of 2.

   Appropriate value for the parameter L: 2**8 Answer: 2^8

   Appropriate value for the parameter B: 14 Answer: 14

   Explanation
   The number of entries in a page map is equal to the number of virtual pages. Since we have a 16 bit virtual address with 2^8 byte pages, that means that our VPN is 8 bits long so there are a total of 2^8 virtual pages and therefore entries in the page map.
   Each page map entry is equal to the width the PPN + 2 for the dirty and resident bits. Since the physical address is 20 bits long, and 8 of those bits are for the offset, then the PPN is 12 bits long. So each page map entry is 14 bits long.

2. If the engineers decide to increase the page size to 512 (2^9) bytes but keep the same size virtual and physical addresses, what affect will the change have on the following architectural parameters?

   Size of page map entry in bits:

   - ○ doubled
   - ○ increased by 1
   - ○ stays the same
   - ◉ decreased by 1 ✔
   - ○ halved

   Number of entries in the page map:

   - ○ doubled
   - ○ increased by 1
   - ○ stays the same
   - ○ decreased by 1
   - ◉ halved ✔

   Number of accesses to page map to translate a single virtual address:

   - ○ doubled

doubled

increased by 1

stays the same ✔

decreased by 1

halved

Explanation
Doubling the page size without changing the size of the virtual or physical addresses results in half as many physical and virtual pages. Half as many physical pages result in 1 fewer bit per PPN so the size of each entry in the page table decreases by 1. The number of entries in the page table is 2^{VPN bits}, so if VPN bits decreases by 1, then total number of entries goes down by a factor of two. The number of accesses to the page map required to translate a single virtual address remains the same.

Back to the micro Beta with 256-byte pages: A test program is run on the micro Beta and halted just before the execution of the two instructions shown below. The first 8 locations of the page table at the time execution was halted are shown in the page map below the code; the least recently used page ("LRU") and next least recently used page ("next LRU") are as indicated. Execution resumes and the following two instructions at locations 0x1FC and 0x200 are executed:

```
        LD(R31, 0x34C, R1)    | PC = 0x1FC
        ST(R1, 0x604, R31)    | PC = 0x200
```

VPN	D	R	PPN
0x0	0	1	0x123
0x1	1	1	0x007
LRU → 0x2	0	1	0x602
0x3	--	0	--
0x4	1	1	0xACE
next LRU → 0x5	1	1	0x097
0x6	1	1	0x790
0x7	--	0	--

3. Assume that all the code and data for handling page faults is located on physical page 0. When the micro Beta executes the two instructions above it will access five different physical pages. Please list the five physical page numbers below **in hex**, in the order they are first accessed. Hint: all the page map information you need is shown in the page table.

Addr 1: 0x 7 Answer: 7

Addr 2: 0x 0 Answer: 0

Addr 3: 0x 602 Answer: 602

Addr 4: 0x 097 Answer: 97

Addr 5: 0x 790 Answer: 790

Explanation
The first memory location to be accessed is the fetch of the LD instruction. This instruction is located at address 0x1FC. Since our pages have 256 bytes in them, that means that the bottom 8 bits of the address are used for the page offset and the top bits are used for the VPN. So we begin by looking up VPN = 0x1. We find this page in our page table with a corresponding PPN of 0x007. This is the first physical page that

We find this page in our page table with a corresponding PPN of 0x007. This is the first physical page that is accessed.

Next we need to fetch the data at 0x34C. This corresponds to VPN = 3 which is not resident in main memory according to our page map. This means that we need to remove the LRU page and use its PPN for VPN 3. The LRU page is page 2, so we mark it as not resident, and now we can make page 3 resident with a PPN of 0x602. Recall that the code that was accessed for handling the page fault is located at physical page 0, so the next two physical pages accessed are 0 and then 0x602.

Now we move on to fetch out ST instruction from address 0x200. This corresponds to VPN = 2 which we just removed from main memory. So once again we get a fault, and now we replace the next LRU page for VPN 2. So VPN 5 has its resident bit set to 0, and VPN 2 has its resident bit set to 1 and its PPN set to 0x97. So these two memory accesses were to PPN 0 and 0x97. Since the question asks about 5 different physical pages being accessed, we don't count page 0 again.

Finally, the ST writes to virtual address 0x604 whose VPN is 6. This page is resident in main memory and is located in PPN 0x790. So the last physical page accessed in this sequence of instructions is 0x790.

Submit

Answers are displayed within the problem

Virtual Memory

0:00 / 0:00 1.0x

Start of transcript. Skip to the end.

Virtual memory allows programs to behave as if they have a larger memory than they actually do.

The way this works is by using virtual addresses, which refer to addresses on disk, in our programs.

The virtual addresses are translated into physical addresses using the page map which is a lookup table that has one entry per virtual page.

The page map knows whether the

Video Download video file

Transcripts Download SubRip (.srt) file Download Text (.txt) file

Discussion

Topic: 16. Virtual Memory / WE16.1

Hide Discussion

Add a Post

Show all posts by recent activity

There are no posts in this topic yet.

Previous Next