Conor Curry
CS 1541
HW #4

PT1						
Memory Accesses	**VA Page**	**T**	**L**	**B**		**Result**
0x05C0	0	C	0	3	4	TLB Hit
0x1033	1	0	3	4	1	TLB Miss
0x84AC	8	3	4	1	8	Page Fault
0x4FE0	4	3	4	1	8	TLB Hit
0xAC0D	A	4	1	8	A	Page Fault
0xB0DE	B	1	8	A	B	Page Fault
0x8004	8	1	8	A	B	TLB Hit
0xF00D	F	8	A	B	F	TLB Miss
PT2						
0x05C0	0	6	0	1	2	TLB Hit
0x1033	0	6	0	1	2	TLB Hit
0x84AC	4	0	1	2	4	Page Fault
0x4FE0	2	0	1	2	4	TLB Hit
0xAC0D	5	1	2	4	5	Page Fault
0xB0DE	5	1	2	4	5	TLB Hit
0x8004	4	1	2	4	5	TLB Hit
0xF00D	7	2	4	5	7	TLB Miss
PT3						
	4KB	8KB				
TLB Misses	2	1				
Page Faults	3	2				
Data Transferred	12KB	16KB				

Conor Curry
CS 1541
HW #4

4:

- **What improved when increasing page size to 8KB, and why did that happen?** Both the TLB miss rate and the Page Fault rate improved. This was because the new page size meant that some pages from the 4KB version were merged into the same page. For example, in the 4KB simulation, memory address 0x05C0 and 0x1033 were on different pages, and had to hit/miss separately. However, in the 8KB version, those two memory locations were both located on page 0. This made it easier to hit with 8KB pages.

- **What got worse when increasing page size to 8KB, and why did that happen?** Even though we were getting hits more frequently, each time we swapped in a page, it meant that we needed to load twice as much data into RAM from disk. This would take twice as long as swaps with 4KB pages. Unfortunately, 8KB pages did not lead to at least a factor of 2 decrease in page faults, so the net effect was that more data was loaded from disk than in the 4KB version.

5: **Do you think these trends would continue if the pages kept getting bigger?**
I think that the trends would continue. While you would make it easier to hit, the page table would continue shrinking, and the TLB would have to do less work, the data transfer overhead from swapping pages would continue to rise as well. The limiting case is when all of RAM is a single page, which means every memory access is a hit, but we loose many of the benefits of having separate pages. This means the larger swaps like we saw in the simulation above, but it also means that we would loose on the memory savings that we can achieve from a mutli-level page table.

Conor Curry
CS 1541
HW #4

Section 2

Section 2						
	0x00403	-	-	-	0x00403=>0x00992	TLB Miss
	0x00404	-	-	0x00403=>0x00992	0x00404=>0x0003A	TLB Miss
	0x00405	-	0x00403=>0x00992	0x00404=>0x0003A	0x00405=>0x00C0B	TLB Miss
	0x00400	0x00403=>0x00992	0x00404=>0x0003A	0x00405=>0x00C0B	0x00400=>0x00D0C	TLB Miss
	0x00401	0x00404=>0x0003A	0x00405=>0x00C0B	0x00400=>0x00D0C	0x00401=>0x00B00	TLB Miss
	0x00405	0x00405=>0x00C0B	0x00400=>0x00D0C	0x00401=>0x00B00	0x00405=>0x00CAB	TLB Hit??
#8						
	0x00403	-	-	-	0x100403=>0x00992	TLB Miss
	0x00404	-	-	0x100403=>0x00992	0x100404=>0x0003A	TLB Miss
	0x00405	-	0x100403=>0x00992	0x100404=>0x0003A	0x100405=>0x00C0B	TLB Miss
	0x00400	0x100403=>0x00992	0x100404=>0x0003A	0x100405=>0x00C0B	0x200400=>0x00D0C	TLB Miss
	0x00401	0x100404=>0x0003A	0x100405=>0x00C0B	0x200400=>0x00D0C	0x200401=>0x00B00	TLB Miss
	0x00405	0x100405=>0x00C0B	0x200400=>0x00D0C	0x200401=>0x00B00	0x200405=>0x00CAB	TLB Miss

7: What Happened on the last access?
We got a “hit” on a virtual address from a different process, so the two had different physical page numbers. I wasn’t sure if I should have replaced the TLB entry or added a new one!
9: What happened on the last access in the second case?

Conor Curry
CS 1541
HW #4

We got a miss on the last access, because we were tagging addresses with process IDs. This eliminated confusion about where to put it and what to call it (miss/hit).
10: process ID tagging in the TLB should help implement shared memory, since we'll be able to see if a process is allowed to share memory with another one. We'll be able to validate each page access that way, by considering the process ID when determining access validity.

Section 3:

11: 2^(32-12) = **2^20 PTEs**
12: 2^20 * 3 bytes = **3 MiB**
13: 2^8 * 4 bytes = 2^10 Bytes, or **1 KiB**
14: 2^12 * 3 bytes = **12 KiB**
15: 2^12 * 4 KiB = **16 MiB**
16: 50 * (1 KiB + 12 KiB) = **650 KiB**