Short Assignment - Page Faulting

The virtual address space for a process (i.e. the operating system's view of the address space, which at the beginning of the following question hasn't yet been reflected in the page tables used by the hardware) is illustrated below, along with the first few instructions of the program that will be executed after it is paged in off of the disk.

# pages	start	end	
3	`00000,000`	`00002,FFF`	Demand-paged from /bin/program pages 0, 1, 2
2	`00030,000`	`00031,FFF`	Demand-allocate (data)
1	`08000,000`	`08000,FFF`	Demand-allocate (stack)

```
00000,FFC    PUSH R0
00001,000    CALL 0x2,000

00002,000    SET R1 = 0x30,000
00002,004    STORE R0 -> *R1
00002,008    HALT
```

Note that a comma is used to separate the page number from the offset, for clarity. Additionally, you may assume the following:

- All instructions are 4 bytes long, and each instruction requires a single 4-byte fetch to retrieve it from memory[1]
- PUSH and CALL operate as in Lab 1, except that (a) 4-byte values are stored to memory, and (b) SP is decremented by 4, not 2

The program starts in the following state:
```
SP = 08001,000 (decrement -4 before use)
PC = 00000,FFC
```

There is a single physical page, 00000, allocated as the page directory (the root of the page table), and all entries are marked non-present. Physical pages 00001 through 00009 are available for allocation, although you may not need to use all of them.

Assume a file is divided into 4096-byte *blocks*, numbered from 0, so the first 4096 bytes of a file is block 0.

(continued on next page)

1 Curious readers might wonder how "CALL 0x2000" and "SET R1 = 0x30000" could be encoded into 32-bit instructions. I'll post something to Piazza about this, rather than complicating the assignment.

For this assignment, you need to list all of the steps that occur from the start of execution until HALT is executed. Each step is one of the following:

- **instruction fetch(address)**: fetch an address from memory at the indicated address
- **attempt(instruction)**: attempt instruction (give text of instruction, i.e., "PUSH #10")
  If the instruction accesses memory, describe the access as follows:
  - **load(address)**: read data from the indicated memory address
  - **store(address)**: write data to memory
- **success**: the instruction attempted in the previous step completed successfully
- **fault(address, type)**: page fault; indicate the fault address and instruction/data load/data store
- **page allocate = xxxxx**: allocate a page; indicate the page number you chose
- **00000[i].page = xxxxx**: set the ith entry of the page directory to point to physical page xxxxx
- **xxxxx[i].page = yyyyy**: set the ith page table entry; xxxxx and yyyyy were previously allocated
- **read_block(F, blknum, yyyyy)**: read block <blknum> from file F into physical page xxxxx
- **return_from_fault**

Note that if a step (fetch, attempt/load, attempt/store) results in a page fault, that step needs to be repeated after the page fault returns.

Submission instructions: Please submit your answer via Canvas, either in PDF format or via the text entry field.