Assumption: used 2 level page tables to implement this

1. Instruction fetch (00000,FFC)
2. Fault(00000,FFC, instruction) // first instruction fetch; PDE[0] and PTE[0] not present
3. Page allocate = 00001 // allocate new page table for low 4 MB region (PDE 0)
4. 00000[0].page = 00001
5. Page allocate = 00002 // allocate physical frame for program pg1
6. 00001[0].page = 00002
7. read_block(/bin/program, 0, 00002)
8. Return_from_fault
9. Instruction fetch (00000,FFC)
10. attempt(PUSH #10)
11. store(08000,FFC)
12. fault(08000,FFC, data store) // stack region not yet mapped
13. Page allocate = 00003 // allocate new page table for stack region (PDE 32)
14. 00000[32].page = 00003
15. Page allocate = 00004 // allocate actual stack frame (physical page)
16. 00003[0].page = 00004
17. Return_from_fault
18. store(08000,FFC)
19. Success
20. Instruction fetch (00001,000)
21. fault(00001,000, instruction) // program page 1 not yet loaded
22. Page allocate = 00005 // allocate new frame for program pg2
23. 00001[1].page = 00005
24. read_block(/bin/program, 1, 00005)
25. Return_from_fault
26. Instruction fetch (00001,000)
27. attempt(CALL 2,000)
28. store(08000,FF8)
29. Success
30. Instruction fetch (00002,000)
31. fault(00002,000, instruction) // program page 2 not yet loaded
32. Page allocate = 00006 // allocate new frame for program pg3
33. 00001[2].page = 00006
34. read_block(/bin/program, 2, 00006)
35. Return_from_fault
36. Instruction fetch (00002,000)
37. attempt(MOV EAX -> *(00010,000))

38. store(00010,000)
39. fault(00010,000, data store) // first access to data region - no PTE[16] entry
40. Page allocate = 00007 // allocate physical frame for data page
41. 00001[16].page = 00007
42. Return_from_fault
43. attempt(MOV EAX -> *(00010,000))
44. Success
45. Instruction fetch (00002,004)
46. attempt(HALT)
47. Success