

MEM1-BB: PAS, UNI-P, MULTI-P, BB: 121 – 150

121 [T / F] The process address space (PAS) is basically the logical addresses where the segments of your process live (e.g. stack, data, heap, code, ...).

122 [T / F] The stack grows from low to high addresses and the heap grows the other way around.

123 [T / F] Separate processes share the same data segment (the data segment is where global variables live).

124 [T / F] All the addresses visible in your process (e.g. pointer address, function address) are all *logical/virtual* address. The OS does not provide any syscall that allows your process to know its actual physical location on DRAM.

125 [T / F] In this class, I should only care about 4 important segments of PAS: stack, heap, data, and code.

mem1-uni/multi-programming:

126 [T / F] Uniprogramming means only one user process exists at a time in the memory.

127 [T / F] To implement uniprogramming, we need to introduce logical/virtual addresses.

128 [T / F] In uniprogramming, context switching (of live processes) is as fast as in multiprogramming.

129 [T / F] In uniprogramming, we likely waste the memory space because a process might not use the entire memory (DRAM).

130 [T / F] During this Thanksgiving I'm grateful that my OS supports multiprogramming – I can have my professor's powerpoint slides, my family's discord channel, and my friend's TikTok video, all opened at the same time, Yeah!!

131 [T / F] Multiprogramming/timesharing allows multiple processes to run on a processor core at (literally) the same time. .

132 [T / F] Multiprogramming/timesharing allows multiple processes to co-exist in the memory at the same time. .

mem1-base-and-bound-(BB)-dynamic-relocation:

133 [T / F] "Dynamic relocation" and "base and bound (BB)" are the same terminologies.

134 [T / F] BB must manage multiple bound values for each process (e.g., the stack bound, the heap bound).

135 [T / F] Internal fragmentation means wasted space inside the PAS; i.e. you are given the memory space but you're not using it.

136 [T / F] External fragmentation means scattered small memory "holes" outside the PASes that cannot be used directly for new processes. B20

137 [T / F] BB suffers from internal fragmentation but not external fragmentation.

138 [T / F] Fast forward today, to solve both internal and external fragmentations, we have to use multi-level page tables, but the professor said it's too much for this course.

139 [T / F] OS developers could implement BB without new hardware support.

140 [T / F] In BB, when referencing a logical address, the resulting physical address is the sum of the logical address and the base address (and the logical address must be within the bound).

141 [T / F] In BB, the MMU cost is expensive because we need to have two more registers and addition and comparator units.

142 [T / F] If the logical address is beyond the bound, MMU will throw a segfault exception, which makes the CPU jumps to the OS' `segfault_exception_handler()`, which in turn will kill the segfaulting process. That's so coooooo! Now I know how segfault works!

143 [T / F] The registers on the CPU (e.g. `eax`, `ebx`, "R1", "R2") are just temporary values, hence they don't need to be saved in the PCB when the process is interrupted and swapped up (during a context switch).

144 [T / F] The OS is like a super nice librarian. If I get kicked out from a study room (interrupted + context switches), the librarian will remember all the positions of my book, bag, chair, pencil, etc. etc. (i.e., the register values) such that when I have the room back, everything will be in the same place as if I was not interrupted.

145 [T / F] In BB, multiple processes of the same program can share the same code segment.

146 [T / F] In BB, we cannot move PAS around, hence the external fragmentation (scattered holes) can never be filled.

147 [T / F] In BB, reshuffling processes in the memory is possible but it's slow because all processes must be stopped and many `memcpy()`s must be done.

148 [T / F] BB is practical because we know the memory consumption of our program ahead of time.

149 [T / F] I promise I'll be ready with hex arithmetic and hex-to-bit conversation and vice versa.

150 [T / F] ... 150