2 Solutions

1.1 Processes

1. Logical control flow: the process executes as if it has complete control over the CPU. The OS implements this by interleaving execution of different processes via context-switching(exceptional control flow...).

2. Private linear address space: the process executes as if it has access to a private contiguous memory the size of the virtual address space.

1.2 Virtual Memory

1. Sharing of a single physical page in multiple virtual address spaces (e.g., shared library code).

2. Memory protection mechanisms (e.g., page-granular read/write/execute permissions or protecting one processs memory from another).

1.3 TLBs

No. The TLB caches page table entries. After a TLB miss, we do an in-memory page table lookup. A page fault occurs if the page table entry is invalid.

1.4 Some Differences between Java and C

1. C allows pointer arithmetic; Java does not.

2. C pointers may point anywhere (including the middles of memory objects); Java references point only to the start of objects.

3. C pointers may be cast arbitrarily (even to non-pointer types); casts of Java references are checked to make sure they are type-safe.

1.5 Structs Solution

Consider the following definition of the struct below, answer the questions regarding the struct.

```
typedef struct data_struct {
  int a;
  char b[3];
  short c;
  void *d;
} data_struct;
```

a) Assume you have an data_struct array of size two, on the stack diagram on the following page please shade in and label the memory blocks for each field of each struct. (You can assume that the first array index starts at element 0x0). You will not use all the space on the diagram

b) What is the total size of this struct?

24 bytes, (17 bytes plus 7 byte of internal padding)

c) Would re-ordering the fields from largest to smallest reduce the size of the struct?

No external fragmentation would still keep the size the same (however remember that you should always order from largest to smallest because it helps more often than not)

d) What would be the assembly code for getting the value of field d out of the struct? Assume that the register %rdi points to the beginning of the struct. Return the value in register %rax.

```
movq 0x10(%rdi), %rax
ret
```

Memory Address	+0	+1	+2	+3	+4	+5	+6	+7
0x00 **(array index 1)→**	a	🟩	🟩	🟩	b[0]	b[1]	b[2]	
0x08	c	🟦						
0x10	d	🟪	🟪	🟪	🟪	🟪	🟪	🟪
0x18 **(array index 2) →**	a	🟩	🟩	🟩	b[0]	b[1]	b[2]	
0x20	c	🟦						
0x28	d	🟪	🟪	🟪	🟪	🟪	🟪	🟪