Description of Memory (On Reset)	Physical Address
UART Bootloader Entry Point	0x00000000
Core 1 Entry Point When read != 0, “mov w30,w0”, where w0 = 0x00000504	0x00000504
Core 2 Entry Point When read != 0, “mov w30,w0”, where w0 = 0x00000508	0x00000508
Core 3 Entry Point When read != 0, “mov w30,w0”, where w0 = 0x0000050C	0x0000050C
Interrupt Vector Handlers Undefined Handler SWI Handler Prefetch Handler Data Handler Unused Handler IRQ Handler FIQ Handler *Note only 256 bytes is allocated for each handler. They should be short because they interrupt normal program flow. If more space is needed then use a branch with link somewhere else in memory.	0x00000800 0x00000800 0x00000900 0x00000A00 0x00000B00 0x00000C00 0x00000D00 0x00000E00
User Code Entry Point Gets filled with user code by bootloader. When bootloader finishes receiving data, branch to this address which contains _start: for user code.	0x00001000
Stack Pointers Core 0 Stack Pointer Core 1 Stack Pointer Core 2 Stack Pointer Core 3 Stack Pointer *Note the stack pointer progresses downwards in memory. It subtracts first, then writes. 4096 bytes of RAM is dedicated for each core’s “stack”.	0x3F000000 0x3F000000 0x3EFFF000 0x3EFFE000 0x3FFFD000
Peripheral Address Base	0x3F000000

Description of Memory (On MMU Enable)	Physical Address
MMU Translation Table Filled with 1024 entries of 1MB sections representing all of the physical memory.	0x00000000
End Of Translation Table Contents: 0x00000000 *Since bits[1:0] != 0b10, MMU knows it’s end of table	0x00000400
Core 1 Entry Point When read != 0, “mov w30,w0”, where w0 = 0x00000504	0x00000504
Core 2 Entry Point When read != 0, “mov w30,w0”, where w0 = 0x00000508	0x00000508
Core 3 Entry Point When read != 0, “mov w30,w0”, where w0 = 0x0000050C	0x0000050C
Interrupt Vector Handlers Undefined Handler SWI Handler Prefetch Handler Data Handler Unused Handler IRQ Handler FIQ Handler *Note only 256 bytes is allocated for each handler. They should be short because they interrupt normal program flow. If more space is needed then use a branch with link somewhere else in memory.	0x00000800 0x00000800 0x00000900 0x00000A00 0x00000B00 0x00000C00 0x00000D00 0x00000E00
User Code Entry Point Contains _start: for the user code. Since user code is already running when MMU is enabled, it will overwrite bootloader to save some space.	0x00001000
Stack Pointers Core 0 Stack Pointer Core 1 Stack Pointer Core 2 Stack Pointer Core 3 Stack Pointer *Note the stack pointer progresses downwards in memory. It subtracts first, then writes. 4096 bytes of RAM is dedicated for each core’s “stack”.	0x3F000000 0x3F000000 0x3EFFF000 0x3EFFE000 0x3FFFD000
Peripheral Address Base	0x3F000000