```
@Done
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@
@ Memtest.s
@
@
@
@
@ Description: Contains code for testing the SRAM and DRAM
@              This code is normally ran before copying the system from ROM to SRAM or DRAM
@
@ Table of Contents:
@   - mem_test: Tests the memory address range that is passed.
@               The test erases anything stored here before.
@
@
@
@
@
@
@ Revision History:
@ Name          Comment                 Date
@ Will Werst    Initial version         Some lonely night around 6/10/17
@ Will Werst    Comment                 October 2017


.include    "at91rm9200.inc"
.include    "system.inc"


.text
.arm

@ mem_test
@
@ Description: Tests memory of passed memory section.
@              In the process of testing memory, this function destroys all data
@              that was in the memory already.
@
@ Operational Description: The code works by writing a sequence of numbers
@                          to successive bytes in the memory that are fairly
@                          random and do not repeat on any 2^n periodicity, and
@                          hence should expose any address line connectivity issues.
@                          A sequence is written to memory, and then read back
@                          and checked against the expected sequence. Then,
@                          another sequence is written to memory, and this is continued
@                          until r3, the incrementer value, overflows. If
@                          the memory does not show any errors, it is assumed
@                          to be good. While this is not an exhaustive memory
@                          test, it is a simple test that can be used to verify
@                          basic memory functionality.
@
@ Arguments: r0 - starting address to test memory integrity
@            r1 - length of data to test memory integrity of
@
@ Return values: r0 - TRUE if success, FALSE if failure
@                r1 - value read from memory
@                r2 - value expected in memory
@                r3 - relative address where error occurred
@
@ Local variables: r0 - base address - unchanged
@                  r1 - length of memory - unchanged
@                  r2 - relative location into memory
@                  r3 - incrementer value used to generate sequences of data to load into memory
@                  r4 - value to load into memory
@
```

```
@ Shared variables: None
@
@ Global Variables: None
@
@ Inputs: None
@
@ Outputs: None
@
@ Error Handling: None
@
@ Algorithms: None
@
@ Data Structures: None
@
@ Limitations: Does not verify that DRAM refresh is working correctly.
@
@ Registers Changed (besides ARM convention r0-r3): None
@
@ Known Bugs: None
@
@ Special notes: None
@
@ Revision History:
@ Name             Comment              Date
@ Will Werst        Initial version     6/22/2017


.global mem_test
mem_test:
    PUSH {r4, r5, lr}
    LDR r3, =0              @Load incrementer value
    LDR r4, =0
mem_test_loop:
    LDR r5, =0x3F35D4B3
    ADDS r3, r5              @Increment incrementer value
    BCS success             @If have used all incrementer values, and thus overflowed, return
    LDR r2, =0              @Load the initial location to load into memory
    PUSH {r4}               @Store current starting value to recover later for when checking
writedata:
    STR r4, [r0, r2]        @Load value into memory
    ADDS r4, r3             @Increment value to load into memory
    SBC r4, r4, #1          @Subtract carry flag so that wrapping occurs at 2^32 + 1, not 2^32
    ADD r2, #4              @Increment the relative location to load into memory
    CMP r2, r1              @Check if written to all locations in memory
    BLT writedata           @If haven't, then keep writing, else go to check memory
    LDR r2, =0              @Reset initial location in memory
    POP {r4}                @Recover starting value and check data in memory
checkdata:
    LDR r5, [r0, r2]        @Load value from memory
    CMP r5, r4
    BNE failure
    ADDS r4, r3             @Increment value to load into memory
    SBC r4, r4, #1          @Subtract carry flag so that wrapping occurs at 2^32 + 1, not 2^32
    ADD r2, #4              @Increment the relative location to load into memory
    CMP r2, r1              @Check if written to all locations in memory
    BLT checkdata
    B mem_test_loop
failure:
    LDR r0, =FALSE
    MOV r1, r5
    MOV r3, r2
    MOV r2, r4
    B mem_test_end
success:
    LDR r0, =TRUE
    @B mem_test_end
```

```
mem_test_end:
    POP {r4, r5, pc}

.end
```