EE234 Spring 2019
Example Final Programming Problems

1. Write a C function that counts the number of bits that are set to a '1' at a 32-bit memory location called "datain", and writes the total to a location called "bitcount".

   ```
   #define   datain      *((uint32_t*) 0xF8000018)
   #define   bitcount    *((uint32_t*) 0xF8000034)
   ```

2. Write an assembly program that counts the number of bits that are set to a '1' in R0, and writes the total to R1. When finished, enter an infinite loop.

3. Write a C function to add 12 consecutive numbers in an array whose first location is stored at a memory location "array", and store the result in "result".

   ```
   #define   array       *((uint32_t*) 0xF8000018)
   #define   result      *((uint32_t*) 0xF8000034)
   ```

4. Write an assembly program to load 12 consecutive words from "array", add them together, and store the results in "result". When finished, enter an infinite loop. The faster your code executes, the more points you get.

   ```
   .text
   .global main
   .equ array, 0x40000000
   .equ result, 0x40001000

   LDR  R0, =array
   LDR  R1, =result
   ```

5. A real-time processing environment produces 20 PWM waveforms, and the pulse-width data for all 20 waveforms is stored in ascending consecutive memory starting at the label "widths". Write a C function that can place the largest width value in memory location "largest", and the address of the largest location in "lagrestaddress".

   ```
   #define   widths          *((uint32_t*) 0xF8000100)
   #define   largest         *((uint32_t*) 0xF8000200)
   #define   largestaddress  *((uint32_t*) 0xF8000200)
   ```

6. Write an assembly program to find the largest of twenty 32-bit values stored in ascending memory starting at "data1". Store the largest data element found at "largest", and store the address of the largest data found at "largest address".

```
.text
.global main
.equ data1, 0x40000000
.equ largest, 0x40001000
.equ largestaddress, 0x40001000

LDR  R0, =data1
LDR  R1, =largest
LDR  R2, =largestaddress
```

7. Write a C function to transfer an arbitrarily sized array from memory to an output port. The array starts at location "startarray", fills ascending memory, and ends with a terminal character (0xFFFFFFFF). The port has a FIFO accessed through a location labelled "port1". Your function should transfer 12 locations at a time to port1, and terminate when the terminal character is encountered. After each 12-location transfer, check the "fifofull" memory location; if it is a '1', terminate the function.

```
#define   startarray      *((uint32_t*) 0x40000000)
#define   port1           *((uint32_t*) 0xF8000000)
#define   fifofull        *((uint32_t*) 0xF8000004)
```

8. Write an assembly subroutine to transfer the contents of "data1" ascending memory starting at address 0x4000000 to a "port1" FIFO at address 0xF8000000. Transfer 12 locations at time, and stop the transfer when you encounter a memory location with all 1's (0xFFFFFFFF). After each 12-location transfer, check the "fifofull" location at 0XF8000004 – if a '1' is in the LSB, stop the transfer and return to the calling program. Observe the ARM calling conventions.

```
.text
.global main
.equ data1, 0x40000000
.equ port1, 0xF8000000
.equ fifofull, 0xF8000004

LDR  R0, =data1
LDR  R1, =port1
LDR  R2, =fifofull
```

Write an assembly program that can scan through a list of 24 numbers stating at location "list" and place the largest number in R7.

General Review Topics

General processor architectures: instruction fetch cycle, decode, data path configuration, source and destination addresses for one, two, and three address machines

Machine (op) codes, immediates, bit-field overloading, program flow redirection, stack, context switching, interrupt processing

ALU status codes

RISC vs. CISC

ARM architecture: Load/Store, addressing modes, GPRs, conditional execution, CPSR, processor modes, interrupts, vector table, AXI bus

ARM programming: instruction formats, assembler directives, code segments, .elf file, initialization, stack, calling conventions

Memory: types/technologies, uses, models, cache, virtual memory, wait states, dynamic wait states, bus grant, DMA

Timers and timing: Main clock, global timer, TTC, watchdog timer

Serial busses: I2S, SPI, UART