EECS 373 Midterm 1 Retry

2 March 2024

40 minutes permitted.

No cellphones, internet, or communicating with others about the exam (except course staff). One double-sided 8.5"×5.5" note sheet is permitted. No other access to course material is permitted.

Name

UM Uniqname

Sign below to acknowledge the Engineering Honor Code: "I have neither given nor received aid on this examination, nor have I concealed a violation of the Honor Code." "Concealed" should be interpreted as "have failed or will fail to report".

Signature

1 Assembly and ABI [20 pts.]

1. [5 pts.] "rand" is an ABI-compliant function with the following prototype: int rand(void). The following ARM assembly language procedure implements the C function "get_random_even".

```
uint32_t get_random_even(void);
1
2 get_random_even:
3
4 bl rand
5
6 lsl r0, r0, #1
7
8 pop {pc}
9
```

   However, this is not ABI-compliant. What additional line is needed for the code to be ABI compliant and correct? Indicate the number of the empty line where you would like to add your code, then the contents of the line you would add.

2. [15 pts.] Figure 2 contains non-ABI compliant ARM assembly language implementation of the C function "push_and_pop_circular_buffer". There are exactly 7 lines of code that need to be modified to make it ABI compliant. No lines need be added or removed. In Figure 1, list which lines of code need to be modified, and their new versions. Assume that we are using the ABI compliant implementation of the function "get_random_even()".

Line number w. error	Corrected version of line

Figure 1: Answers

```
void push_and_pop_circular_buffer(uint32_t* buffer, int buffer_length,
int* buffer_index) {
// index goes in r3
  int index = *buffer_index;
// end goes in r1
  uint32_t* end = buffer + buffer_length;
// buffer stays in r0
  buffer += index;
// ret goes in r4
  uint32_t ret = *buffer;
// insert goes in r7
  uint32_t insert = get_random_even();
  *buffer = insert;
  buffer += 1;
// next_buffer_index goes in r5
  int next_buffer_index = index + 1;
  if (buffer == end) {
    next_buffer_index = 0;
  }
  *buffer_index = next_buffer_index;
  return ret;
}

Note: ; is a line-comment character for the ARM assembler you are using.

1.  push_and_pop_circular_buffer:
2.    PUSH {lr}
3.    PUSH {r4-r7}
4.    LDR r3, [r2] ; r3 is index
5.    ADD r1, r0, r1 ;r1 = buffer + buffer_length * size_of(uint32_t)
6.    ADD r0, r0, r3 ; r0 = buffer + index * size_of(uint32_t)
7.    LDR r4, [r0] ; r4 = ret
8.    PUSH {r0-r2}
9.    BL get_random_even
10.   MOV r7, r0 ; r7 = get_random_even()
11.   POP {r0-r2}
12.   STR r7, [r0, #4] ; *buffer = insert; buffer += 1;
13.   ADD r5, r3, #1 ; r5 = next_buffer_index
14.   CMP r0, r1 ; if (buffer == end)
15.   BEQ skip
16.   MOV r5, #0 ; next_buffer_index = 0
17. skip:
18.   STR r5, [r2] ; *buffer_index = next_buffer_index;
19.   MOV r0, r4
20.   POP {r4-r7}
21,   POP {lr}
```

Figure 2: C and assembly source code.

2 Build Process [10 pts.]

1. [2 pts.] Use at most one sentence to indicate why one might want to "strip symbols" from an executable.

2. [2 pts.] Should symbols be stripped before or after linking?
   - ◯ Must be before.
   - ◯ Must be after.
   - ◯ Either before or after will work.

3. [2 pts.] Mark all the file types that objdump will accept as input when given the "-S" (disassemble) flag.
   - ◯ Assembly.
   - ◯ C.
   - ◯ C++.
   - ◯ Linker scripts.
   - ◯ Object files.
   - ◯ Executables.

4. [2 pts.] Indicate all reasons for using "make", i.e., the things that using "make" enables that would not be practical without it.
   - ◯ Enabling access to earlier versions of source files.
   - ◯ Avoiding unnecessary assembler, compiler, and linker runs.
   - ◯ Automating the build process.
   - ◯ Enabling debugging of executables.
   - ◯ Providing a graphical user interface to manage builds.

5. [2 pts.] You write a valid ARM assembly file, assemble it, then disassemble the object file. The dissassembled version differs from the version you wrote, and the differences aren't limited to comments. Use at most two sentences to indicate a type of difference that might occur.

3 [10 pts.] Interrupts

(a) [2 pts.] Which of the following maps interrupt numbers to ISR addresses?

- ◯ NVIC
- ◯ MMIO
- ◯ APB bus
- ◯ AHB bus
- ◯ EXTI controller

(b) [5 pts.] A single interrupt occurs during the execution of a process. Label the following 1–5 in the order in which they happen. 1 is what happens first and 5 is what happens last. One blank will have two numbers in it.

- • [ ] Interrupt pending bit goes high.
- • [ ] External event in the world.
- • [ ] Executing in ISR mode.
- • [ ] Executing in thread mode.

(c) [3 pts.] Use at most three sentences to describe a scenario in which tail chaining happens. Be specific and include relative times of when events happen. How does tail chaining improve performance of a system?

4 Timers [10 pts.]

1. [3 pts.] Which register holds the value that indicates how to divide the CPU's clock frequency in order to clock the counter? Choose one.

   ◯ CCR.

   ◯ ARR.

   ◯ PSC.

2. [3 pts.] If setting a PWM timer, what expression can be used to determine the duty cycle? The variables you can use are CCR, ARR, PSC (prescaler), and f_clock (clock frequency).

3. [4 pts.] To produce a 75% duty cycle signal with a period of 1 ms, find a CCR, ARR, and PSC if the clock frequency is 1 MHz. Assume 8-bit registers.

   CCR:

   PSC:

   ARR:

This page may be used for work. Please hand it in with the exam and reference it from the associated questions if you would like it to be considered when determining partial credit.