EECS 373 Midterm 1
Winter 2022

10 February 2022

No calculators or reference material.

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."

Signature

1 ABI (7 pts.)

```
void foo(int i, int adjustment, char z)

foo:
  push {r3, r4, r6, LR}
  mov r6, r1
  add r4, r0, #5
  mov r5, r2
loop:
  cmp r0, r4
  beq end
  push {r0}
  sub r2, r5, r6
  mov r0, r2
  bl print
  pop {r0}
  add r0, #1
  b loop
end:
  pop {r3, r4, r6, PC}
```

You are conducting a code review of a function, "foo", from Team Apple, one of two development teams you manage. It calls an ABI compliant "print" from Team B. The function "foo" has arguments, two integers and a character, and has no outputs. It prints the character, adjusted by the second integer argument, a total of 5 times. The function "print" takes a single character as an input and has no outputs. The "print" function has already been validated, and while Team Apple assures you that their function is functionally correct, they have a history of introducing defects into shipped products. Your job is to determine whether the "foo" function is a valid ABI-compliant function and to justify your answer.

Is it a valid, ABI-compliant function?

◯ Yes. ◯ No.

Justify in one terse sentence.

2 Assembly and memory layout (10 pts.)

The Table 1 represent two sections of memory in a Cortex-M processor. For convenience the memory values in the address 0x08000XXX block have been decoded into instructions. Label locations are indicated in italics. Assume that the Program Counter (PC) is 0x08000104 and the Stack Pointer is 0x20000000. Determine the value of R0, R1, R2, and the values of the memory addresses in the right table (i.e., 0x20000000 to 0x1FFFFFD0) when the PC = 0x08000114. Leave any unknown memory values blank.

Table 1: Memory and Register Contents

Address	**Instruction**	**Address / Register**	**Value**
0x08000104	MOV R0, #3	0x20000000	
0x08000108	MOV R1, #7	0x1FFFFFFC	
0x0800010C	MOV R2, #3	0x1FFFFFF8	
0x08000110	BL funA	0x1FFFFFF4	
0x08000114	*done:* B done	0x1FFFFFF0	
0x08000118	*funA:* PUSH {R0,R1,R2,LR}	0x1FFFFFEC	
0x0800011C	*loop:* CMP R0, #0	0x1FFFFFE8	
0x08000120	BEQ next	0x1FFFFFE4	
0x08000124	POP {R0}	0x1FFFFFE0	
0x08000128	BL funB	0x1FFFFFDC	
0x0800012C	PUSH {R0}	0x1FFFFFD8	
0x08000130	B loop	0x1FFFFFD4	
0x08000134	*next:* MOV R0, #6	0x1FFFFFD0	
0x08000138	*loop2:* CMP R0, #0		
0x0800013C	BEQ endA	R0	
0x08000140	POP {R1}	R1	
0x08000144	SUB R0, #1	R2	
0x08000148	B loop2		
0x0800014C	*endA:* POP {R0,R1,R2,PC}		
0x08000150	*funB:* POP {R1,R2}		
0x08000154	PUSH {R1}		
0x08000158	PUSH {R2}		
0x0800015C	PUSH {R0}		
0x08000160	SUB R0, R1, #1		
0x08000164	BX LR		

3 Interrupts (10 pts.)

1. Use at most two sentences to describe the main purpose of the Nested Vector Interrupt Controller (NVIC).

2. Use at most two sentences to explain why it is good practice for interrupt handlers to terminate quickly.

3. On a Cortex M4 processor there are two IRQ handling routines, named IRQ A and IRQ B. IRQ A is set to trigger when signal X transitions from low to high, and has an execution time of 1 clock cycle. IRQ B is set to trigger when signal Y transitions from low to high, and has an execution time of 2 clock cycles. Each of these interrupt handlers sets GPIO A and B, respectively, to HIGH at the start of execution and LOW and the end of execution. Given the following waveforms for X and Y, draw the waveforms for GPIO A and GPIO B given the following preemption priority assignments, where a lower number indicates a higher priority. Assume the processor uses tail chaining. Appended are some excerpts from the ARMv7 Architecture Reference Manual that may be useful. Provide your answers in Figures 1 and 2.

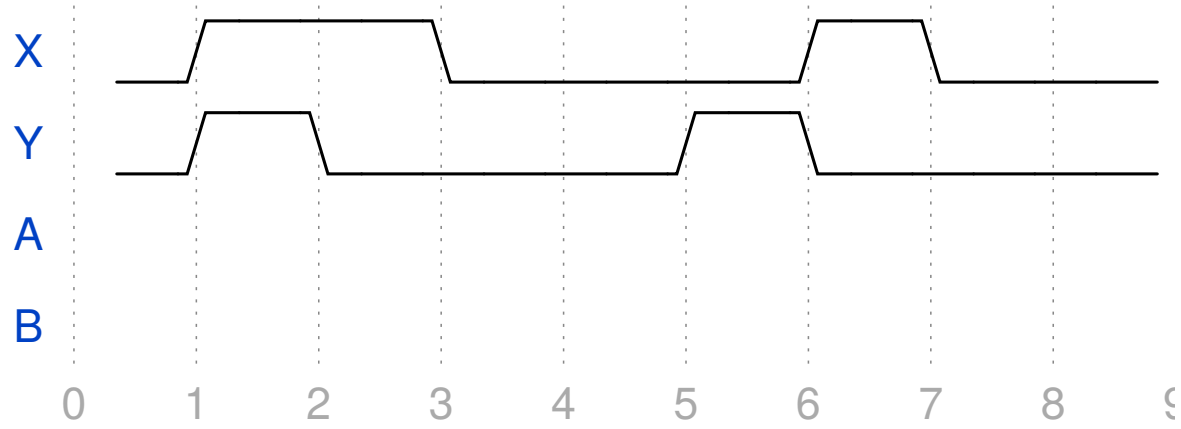

Figure 1: Timing diagram in which GPIO A has a preempt priority of 0 and a subpriority of 0 and GPIO B has a preempt priority of 0 and a subpriority of 1.

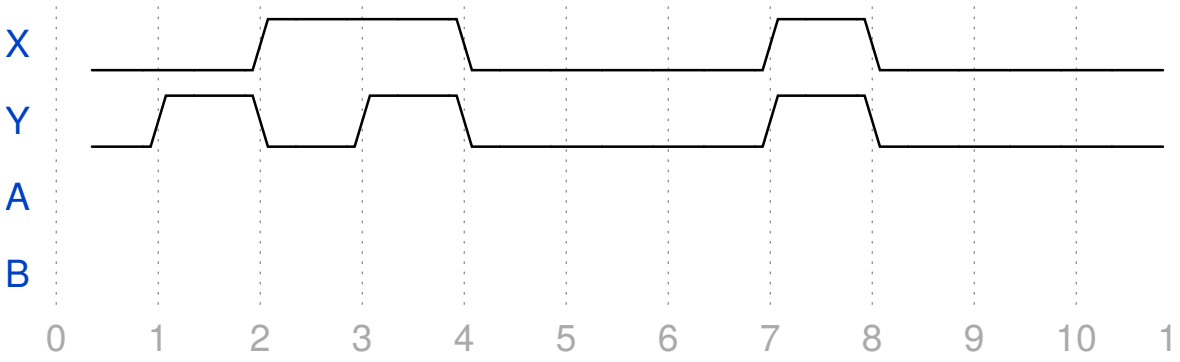

Figure 2: Timing diagram in which GPIO A has a preempt priority of 0 and a subpriority of 1 and GPIO B has a preempt priority of 1 and a subpriority of 0.

4 Build process (8 pts.)

Add directed edges between nodes in Figure 3 to illustrate the flow of information, i.e., data or metadata such as file modification times, in the standard embedded system build process. Don't include edges for command executions, e.g., "Make" should not have an arc to "Linker". We have added a few correct edges to help you get started.

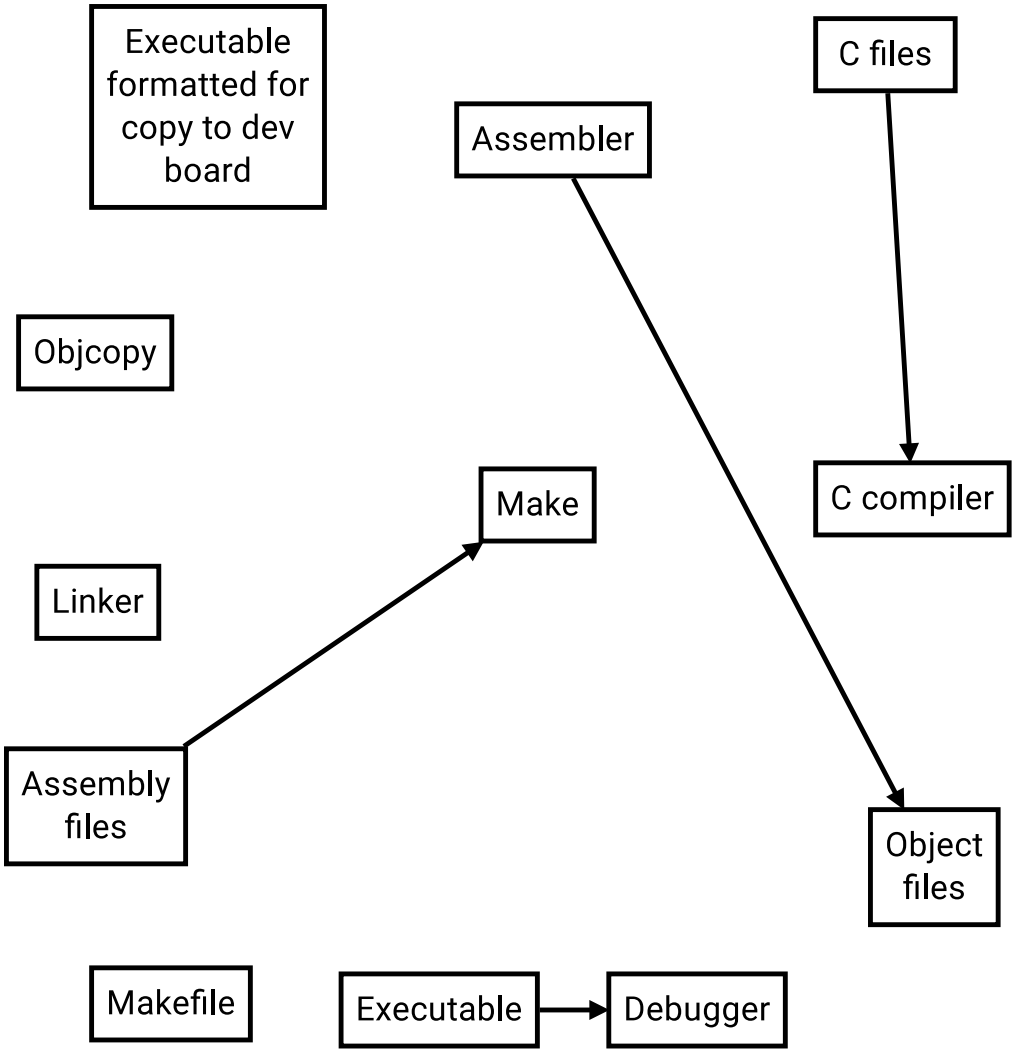

Figure 3: Incompletely specified embedded development board build process.

5 Debugging (4 pts.)

You have implemented an ISR to handle a button press interrupt. The ISR uses a persistent (static) variable in memory to track the state of an LED; it inverts (!=) that variable and writes the new value to an LED interface on the APB, turning on or off the LED. However, on boot the LED is off and pressing the button never turns the LED on, or off. What is the first thing you would do with a debugger to determine whether the ISR is executing? Use at most one sentence.

6 Open-collector style buses and voltage division (5 pts.)

You have implemented an open-collector style bus. In your system, V_{DD} is 3 V. By convention, voltages below 1 V signify activity and voltages above 2 V signify inactivity. Several components are connected to a bus line, each controlling a transistor that electrically connects the line to ground when activated. However, by using a multimeter, you find that when a single component's transistor is turned on, the line voltage decreases to 1.5 V and never drops below 1 V. Your component interface drivers have output resistances of 1 kΩ.

1. What is the pull-up resistance on the bus line?

2. What pull-up resistance would enable an active device to reduce the voltage to 0.5 V.

7 MMIO and logic (4 pts.)

You are designing an APB interface for an ultra-low-power device that supports reads from the following MMIO addresses: 0x0, 0x5, 0x6, 0x7, and 0xd. For each address, indicate the **minimal** number of bits that must be used as inputs to an AND gate used for detecting an access to that address. Do not consider sharing AND gate logic: each address gets its own AND gate. Do not assume that only the lowest-order bits are used: you may skip bits. You needn't consider aliasing with other devices because PSEL can handle that. Consider aliasing among the device's own addresses. You must show your work to receive credit.

1. 0x0
   ◯ 0 ◯ 1 ◯ 2 ◯ 3 ◯ 4 ◯ 5 ◯ 8
2. 0x5
   ◯ 0 ◯ 1 ◯ 2 ◯ 3 ◯ 4 ◯ 5 ◯ 8
3. 0x6
   ◯ 0 ◯ 1 ◯ 2 ◯ 3 ◯ 4 ◯ 5 ◯ 8
4. 0x7
   ◯ 0 ◯ 1 ◯ 2 ◯ 3 ◯ 4 ◯ 5 ◯ 8
5. 0xd
   ◯ 0 ◯ 1 ◯ 2 ◯ 3 ◯ 4 ◯ 5 ◯ 8

8 Cat (1 pt.)

Indicate the concept Figure 4 represents.

- ◯ Non-volatile memory.
- ◯ The customer's view of me.
- ◯ The customer's view of my early-stage product idea.
- ◯ My lab partner.
- ◯ The APB.

Figure 4: An illustration of...

Overflow space. We won't look at this space unless you tell us to after the relevant question.

NVIC operation

Armv7-M supports level-sensitive and pulse-sensitive interrupt behavior. This means that both level-sensitive and pulse-sensitive interrupts can be handled. Pulse interrupt sources must be held long enough to be sampled reliably by the processor clock to ensure they are latched and become pending. A subsequent pulse can add the pending state to an active interrupt, making the status of the interrupt active and pending. However, multiple pulses that occur during the active period only register as a single event for interrupt scheduling.

B1.3.2 Exceptions

Each exception has:

- An exception number.
- A priority level.
- A vector in memory that defines the entry point for execution on taking the exception. The value held in a vector is the address of the entry point of the exception handler, or *Interrupt Service Routine* (ISR), for the corresponding exception.

An exception, other than reset, has the following possible states:

Inactive An exception that is not pending or active.

Pending An exception that has been generated, but that the processor has not yet started processing. An exception is generated when the corresponding exception event occurs.

Active An exception for which the processor has started executing a corresponding exception handler, but has not returned from that handler. The handler for an active exception is either running or preempted by the handler for a higher priority exception.

Active and pending

One instance of the exception is active, and a second instance of the exception is pending.

Only asynchronous exceptions can be active and pending. Any synchronous exception is either inactive, pending, or active.

Exception return

The processor executes the exception handler in Handler mode, and returns from the handler. On exception return:

- If the exception state is active and pending:
  - — If the exception has sufficient priority, it becomes active and the processor reenters the exception handler.
  - — Otherwise, it becomes pending.
- If the exception state is active it becomes inactive.
- The processor restores the information that it stacked on exception entry.
- If the code that was preempted by the exception handler was running in Thread mode the processor changes to Thread mode.
- The processor resumes execution of the code that was preempted by the exception handler.

The Exception Return Link, a value stored in the link register on exception entry, determines the target of the exception return.

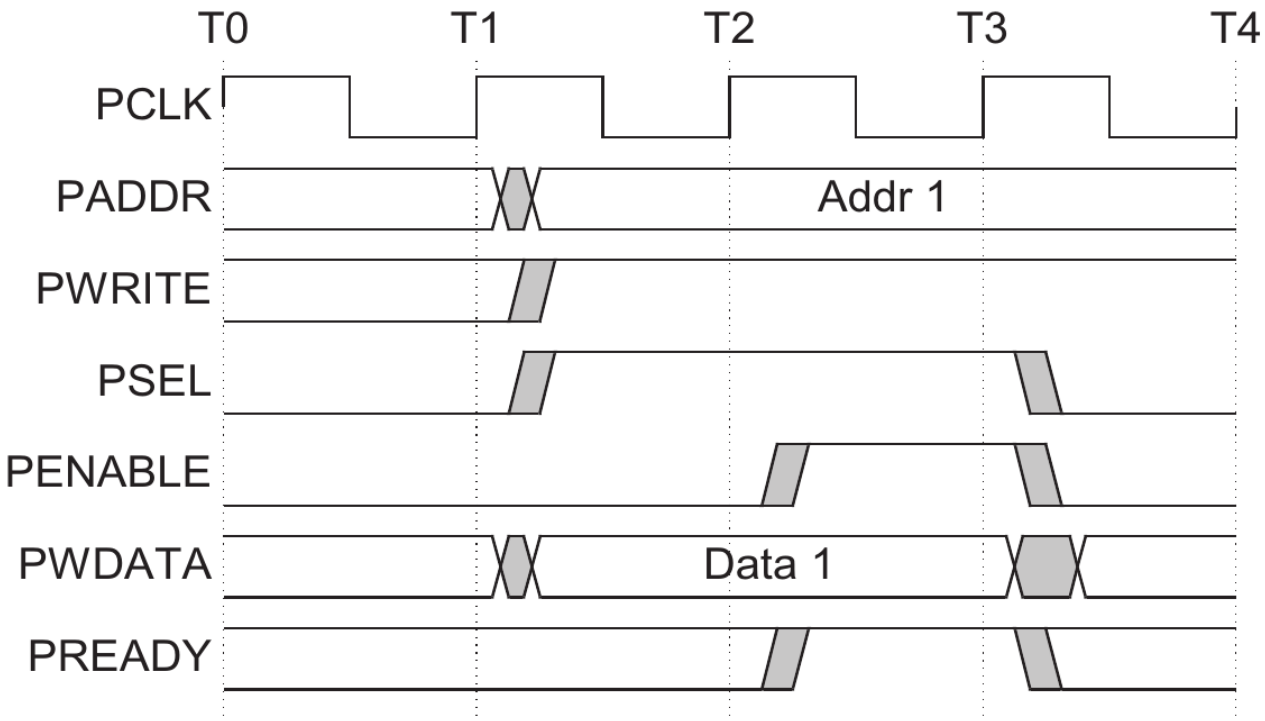

Figure 3-1 Write transfer with no wait states