name:	Vince Rothenberg
course:	csci 10
assignment:	homework 14
prepared:	Tue, May 12, 2020 // 9:12 am

1. What is the stack, and what is it used for?

The Stack is a memory region within the program/process. This part of the memory gets allocated when a process is created. The Stack is used for storing temporary data such as local variables of some function, environment variables which helps us to transition between the functions, etc. It is a highly efficient data structure, push and pop are constant time operations.

2. What are the two (2) principal operations of a stack?

The two primary operations of the stack are PUSH (add data to the stack) and POP (remove data from the stack) instructions. Load And Store PUSH and POP are aliases to some other memory related instructions rather than real instructions, but PUSH and POP are used for simplicity reasons.

3. Write the GNU AS instructions to (1) push r0 onto the stack and (2) pop the stack into r0.

```
push {r0} // copy the value in r0 to the top of the stack

// instructions

pop {r0} // copy the value from the top of stack into r0
```

4. Write the GNU AS instructions to (1) push r0, r1, and r2 onto the stack and (2) pop the stack into r0, r1, and r2.

```
push {r0,r1,r2} OR push {r0-r2}  // push r0, then r1, then r2
pop {r0,r1,r2} OR pop {r0-r2} // pop into r2, then into r1, then r0
```

5. In systems programming, what is an interrupt?

An interrupt is an input signal to the processor indicating an event that needs immediate attention. An interrupt signal alerts the processor and serves as a request for the processor to interrupt the currently executing code, so that the event can be processed in a timely manner. Interrupts are contrasted against polling, the latter of which constantly checks if an event has occurred (such as a button press) in order to execute certain instructions.

6. How does a CPU respond to an interrupt, and what happens after the processor has handled the interrupt?

The processor responds by suspending its current activities, saving its state, and executing a function called an interrupt handler (or an interrupt service routine, ISR) to deal with the event. This interruption is temporary, and, unless the interrupt indicates a fatal error, the processor resumes normal activities after the interrupt handler finishes.

7. Describe hardware interrupts, and how they are typically implemented.

A hardware interrupt request (IRQ) is an electronic signal issued by a hardware device external to the processor (such as a button), to communicate that the device needs attention from the operating system (OS) or, if there is no OS, from the "bare-metal" program (such as ARM code) running on the CPU.

Such external devices may be part of the computer (e.g., disk controller) or they may be external peripherals. Unlike software interrupts, hardware interrupts can arrive asynchronously with respect to the processor clock, and at any time during instruction execution. Consequently, all hardware interrupt signals are conditioned by synchronizing them to the processor clock, and acted upon only at instruction execution boundaries.

8. Provide an example of a hardware interrupt.

Pressing a keyboard key or moving a mouse triggers hardware interrupts that cause the processor to read the keystroke or mouse position.

9. Describe software interrupts.

A software interrupt is requested by the processor itself upon executing particular instructions or when certain conditions are met. Every software interrupt signal is associated with a particular interrupt handler.

A software interrupt may be intentionally caused by executing a special instruction which, by design, invokes an interrupt when executed. Such instructions function similarly to subroutine calls and are used for a variety of purposes, such as requesting operating system services and interacting with device drivers (e.g., to read or write storage media).

Software interrupts may also be unexpectedly triggered by program execution errors. These interrupts typically are called traps or exceptions.

10. Provide an example of a software interrupt.

A divide-by-zero exception will be "thrown" (a software interrupt is requested) if the processor executes a divide instruction with divisor equal to zero. Typically, the operating system will catch and handle this exception. Another example is requesting operating system services and interacting with device drivers (e.g., to read or write storage media).