CPSC 3300 Final Exam

Jonathan Elder

(1) ① They are more accurate, with a higher level of precision than having a software delay. This will make sure also factors like time elapsed will be more consistent through clock cycles.

② There is much better/lower power consumption. When using "busy" loops, it requires the CPU to always be at work.

(2) A) Program-controlled (busy-wait) I/O: Much simplier to implement. This would be used in devices where response time is not important.

B) Interrupt-driven I/O: During an interrupt the CPU is able to go to other task and perform those at the same time. This in turn will make the overall system performance better. Makes it to where the ~~~~ CPU can ~~~~ prioritize I/O operations based on urgency. Although these are much more involved to implement. Leads to more hardware needed which could also mean more overhead personell.

C) Direct memory access: The biggest and most needed advantage is the ability to unload "work" to the CPU. This also offers high speed data transfer capabilities. This has the ability to deal with large sums of data. Again the disadvantage is the fact this adds complexity and careful structures to manage properly.

(3) UART : ASYNC , P2P
I2C : SYNC , BUS
SPI : SYNC , P2P
USB : SYNC , BUS

(4) Operation 1: Enable interrupt in the peripheral module "m".

* When the desired result or event occur then an interrupt signal should be generated

Operation 2: Enable interrupt in the NVIC

* The "Nested Vectored Interrupt Controller" is what manages interrupts in general within our microcontroller. By setting the bit that goes with the interrupt bit from the ISER

Operation 3: Enable the global interrupts found in the CPU.

* By setting the correct bit within the CPU's status register. The "interrupt mask" accomplishes this.

(5) r0 = 0x20000000
r1 = 0xFEDCBA98 r0 = 0x20000004
r0 = 0x20000008
r2 = 0x00000009 r0 = 0x20000008
r3 = 0xFFFFFFFB r0 = 0x20000008
r4 = 0x00000004
r5 = 0x00005678 r0 = 20000000C

(6)

```
        .syntax unified

        .global Max_and_Avg

Max_andAvg:

        POP{r4}        // address for avg
        POP{r3}        // address for max
        POP{r2}        // size of array
        POP{r1}        // address for array

        movs r5, #0    // current index
        LDR r6, [r1]   // max value
        movs r7, #0    // sum of elements

Loop:
        LDR r0,[r1, r5, LSL#2]
        CMP r6, r0
        IT LT
        MOVLT r6, r0   // update Max value if r0>r6
        ADDS r7, r0    // update sum
        ADDS r5, #1    // increment index
        CMP r5, r2
        BNE Loop       // Repeat if r5 is not ≠ r2

        STR r6, [r3]      // store max (was not needed)
        SDIV r0, r7, r2   // Calculates average
        STR r0, [r4]      // store average (was not needed)

Here    b. here        //
```

(7)

```
.syntax unified
.global Filter

Filter:

    PUSH {r4, r5, r6, LR}   // Puts on top of stack
                               to save contents

    LDR r4, =y    // address for y
    LDR r5, =x    // address for x

    LDR r0, [r5]        //  = x0
    LDR r1, [r5, #4]    //  = x1 w/ offset
    LDR r2, [r5, #8]    //  = x2 w/ offset

    LDR r6, [r4]     // value of y
    LSL r0, r0, #3
    ASR r1, #2       // r1 = x1 / 4
    // Calculate formula
    ADDS r6, r0
    SUBS r6, r1
    ADDS r6, r2
    STR  r6, [r4]       // updates value of y
    STR  r1, [r5, #8]   // stores w/ offset to x1
    STR  r0, [r4, #4]   // stores w/ offset to x2

    POP {r4, r5, r6, LR}  // restore the original
                             values in these registers

    BX LR
```

8) A) Chip 1: Rom since it has /RD (Read Enabled w/ active low)

Chip 2: SRAM

Chip 3: SRAM since it has both WE (Write Enabled) and /WR

B) Chip 1: 8192 or 2^{13}, 8 kilobytes

Chip 2: 4096 bytes or 2^{12}, 4 kilobytes

Chip 3: 2048 bytes or 2^{11}, 2 kilobytes

C) Chip 1: 0x0000

Chip 2: 0x2000

Chip 3: 0x3000