CS 220 Spring 22 Exam 2 **Name:____**

Solution

For partial credit make sure to show all work where appropriate. Some answers are better than other answers. Full credit for the best answer.

1. [5] What is the clock cycle time for a processor that has a 2.4 GHz clock. Express your answer in nanoseconds (ns).

   1/2.4e9 = .417e-9 sec = .416 ns.

2. [5] What is the clock rate (frequency) for a processor that has a cycle time of .67 ns (nanoseconds)?

   1/.67ns = 1/.67e-9 sec = 1.5 GHz.

3. [3] What does the command `mkdir ../hw1` do?

   Create a directory named `hw1` in the parent directory.

4. [3] What does the command `mv ./main.c ..` do?

   Move the file `main.c` in the current directory to the parent directory.

5. [10] In the table provided, trace the values of `whatdoido(11)` by filling in the table on the right.

```
whatdoido:
    mov     r1, #0
while:
    cmp     r0, #0
    beq     endwhile
    and     r2, r0, #1
    lsr     r0, r0, #1
    add     r1, r1, r2
    b while
endwhile:
    mov     r0, r1
    bx      lr
```

r0	r1	r2
11	0	1
5	1	1
2	2	0
1	2	1
0	3	
3		

This function counts and returns the number of ones in the binary representation of the argument.

6. Answer questions about the object dump provided. I drew lines to separate the columns. This is an object dump of my solution to the study question where you had to write **`f`**, **`g`**, and **`h`**.

   a. [3] In one short sentence explain what information is provided in the first column.

      These are instruction addresses.

   b. [3] In one short sentence explain what information is provided in the second column.

      Machine code for the instruction at the address in the first column.

   c. [3] In one short sentence explain what information is provided in the second column.

      The corresponding assembly language for the machine code in the second column.

   d. [3] What is the value in the link register **`lr`** immediately after **`f`** is called?

      `0x1032c`

   e. [3] What is the value in **`lr`** the second time **`h`** is called?

      `0x10448` **– I meant for this to mean the second time h is called while the program is running, not the second time it is called textually in the object dump.   So I'll accept 0x10468 as well.**

   f. [3] What value is passed to **`h`** the first time **`h`** is called?

      2

Solution

7. Consider the C variable declarations below

```
int a, *p, *t;
p = &a;
a = 23;
```

   a. [3] What would be printed by **printf("%d\n", *p);**

      23

   b. [3] What would be the effect of the statement **`*t = a;`**

      Crash, segmentation fault, because `t` does not point to a valid address.

   c. [3] What is the type of **`p`**?
      pointer to an integer

   d. [3] What is the type of **`&t`**?

      pointer to a pointer to an integer.

   e. [3] What is the type of **`*&a`**?

      `int`

8. [30] Programming problem. Write an ARM assembly function digitsum that returns the sum of the digits of an unsigned integer passed to it. For example **`digitsum(123)`** would return 6 because 1 + 2 + 3 is 6.
   a. Create a directory **`exam2`** in your repo and put the file **`digitsum.s`** in it.
   b. Write a **`main.c`** that takes a command line argument and calls **`digitsum`** with the argument.
   c. push your files to your repo but be careful and make sure to pull first!
   d. Verify that your files were successfully pushed by going to github.com and making sure they are there.

`SEE NEXT PAGE FOR MY SOLUTION`

Solution

```
// main.c
#include <stdio.h>
#include <stdlib.h>
extern unsigned int digitsum(unsigned int n);

int main(int argc, char *argv[]) {
    printf("%u\n", digitsum(atoi(argv[1])));
}
```

```
// digitsum.s
.cpu cortex-a53
.global digitsum
.text

// compute r0 % r1
mod:
    sdiv r2, r0, r1
    mul  r2, r2, r1
    sub  r0, r0, r2
    bx lr



// sum base-ten digits in r0
digitsum:
    push { r4-r6, lr }
    mov r4, r0  // save r0 because call mod
    mov r5, #0  // the digit sum
    mov r6, #10 // constant 10

while:
    cmp r4, #0       // while (n != 0)
    beq endwhile
    mov r0, r4       // set up call to mod
    mov r1, #10
    bl mod           // n % 10
    add r5, r5, r0   // sum = sum + r0
    sdiv r4, r4, r6  // n = n / 10
    b while
endwhile:
    mov r0, r5
    pop { r4-r6, pc }
```

Solution