Exam 1 Metacognition

Question 1:

My answer: The 5 stage pipeline design is less complex.

Actual answer: The 5 stage pipeline will execute more instructions per second.

Explanation: The 5 Stage pipeline executes up to 5 instructions per clock cycle, while the 3 Stage pipeline can only execute 3 instructions per clock cycle. So the 5 stage pipeline executes more instructions per second.

Why I got it wrong: I didn't study what a pipeline was. I don't remember reading about pipelines or going over it in lecture, because I didn't study the problem I made a guess on the answer.

Question 2:

My answer: It advances the current location counter by 6 bytes if it is not on a 6 byte boundary.

Actual Answer: It advances the current location counter by 64 bytes if it is not on a 64 byte boundary

Explanation: The .align directive takes the value of 6 and multiplies to the power of 2, so the .align will be $2^6 = 64$ bytes. .align takes that value and moves location counter by 64 which will be the next 64 byte boundary if it is not already on that

boundary.

Why I got it wrong! I forget the .align directive takes the value and takes it to the power of 2.

Question 4!

My answer: 128, 63

Correct answer: 127, 63

Explanation

unsigned:

$1111111 = 2^6+2^5+2^4+2^3+2^2+2^1+2^0 = 127$

signed

$01111111 = 2^5+2^4+2^3+2^2+2^1+2^0 = 63$

Why I got it wrong! Math error, I must have typed the value for an unsigned number into my calculator wrong.

Question 11:

My answer: 16

correct answer: 15

Explanation: $2^{15} = 2^{10} \cdot 2^5 = 32 \cdot 1024 > 32,001$

Why I got it wrong: I forgot that a kilobyte is actually 1024 bytes, not just 1000 bytes as it is often generalized as. Since there was 32,001 I thought we would need 16 because $1000 * 32 = 32{,}000 < 32{,}001$.

Question 16:

My Answer: It is the only way to get a full word of data into a register.

Correct Answer: It is the only way to get a full word address into the register.

Explanation: opcodes are only 32 bits so in order to get a full word address into a register the processor has to perform some tricks in order to do so.

Why I got it wrong: I was under the impression that we established addressability so we could access word sized memory. Ex: ldr r4 [r3, A-DSECT] loads a word from memory. We established addressability to DSECT so we could perform that load instruction.

Question 17:

My Answer: ldr r3, =P

Correct Answer: ldr r3, =P

Explanation: In order to establish addressability the correct syntax is ldr rd, = Label.

Why I got it wrong! I didn't realize the p was lower case.

Question 29:

My Answer: $R_1 = 93253687$

Correct Answer: $R_1 = 0X\ 142000$

Explanation: ldr r1, r8, #16

16 = 4 hex bytes

r8 = 0x20000014

$r_1 = 142000$

Why I got it wrong: I didn't understand the output of gdb info regs correctly. I thought each register corresponded to a place in memory. I did not realize that the hex values were actually the value of the registers. I understood how to do the operation correctly, I just used the wrong numbers.

Question 30:

My Answer: $R_7 = 8$

Correct Answer: Memory Address $0X\ 20000010 = 4$

Explanation: Strb r7, [r5; #8]

r5 = 0x20000008
+ 8
= 0x20000010

r7 = 0x20000004

Least Significant byte = 4

So Strb stores 4 at the address 0x20000010

Why I got it wrong! Same as last problem, I tried using the byte in memory located at 0x20000004. The answer would have been 0x de but this wasn't an option. I was running out of time so I guessed.

Question 31:

my answer: R3 = 53681195

Correct answer: R3 = 0x2000011C

Explanation: r6 = 0x2000001C

r3 = r6 + 256 -> 256 = 0x100

0x2000001C
+ 100
0x2000011C

Why I got it wrong! Similar to 29. I didn't understand output of info regs so I assumed r6's address was 0x2000001C and its value was 536870940. obviously I now realize

that a register does not store both memory addresses and a value so my answer was obviously wrong. I added 256