Assignment #2
Maneesh Wijewardhana (1125828)

2021/09/24
TA: Muhammad Rehman

1. Using all registers, there are 8 32 bit registers which is 256 bits. In bytes this is 32 bytes. for words, 16 bits so 512 bits in total which is 64 bytes.

2. These suffixes are called size indicators and are used to perform different operations of different word sizes. For example, MOVE.B would perform the MOVE operand for a byte in a given register. .W would use a word and .L would copy all 4 bytes.

3. -128 (byte, word, longword) (8 bits)
347689 (longword) (19 bits)

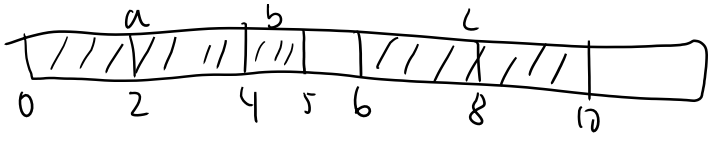
4. $ signifies the immediate value in hexadecimal. Therefore, $123 and 123 are two different things where $123 is in hexadecimal but 123 is in decimal.

5. MOVE.B 123, D4 copies the decimal value 123 into the register D4. However, MOVE.B #123, D4 says that the decimal value 123 is CONSTANT and in memory and gets copied to register D4.

6. MOVE.W #-32769, D1 is wrong because the constant -32769 is out of range for a word. To fix this, using MOVE.L would suffice.
SUB D1, #$15 is also wrong because since the hex value 15 is constant, we cannot subtract the register D1 from it. Also, the source cannot be memory address if its a constant. To fix this, removing the "#" or switching the source → destination would suffice.

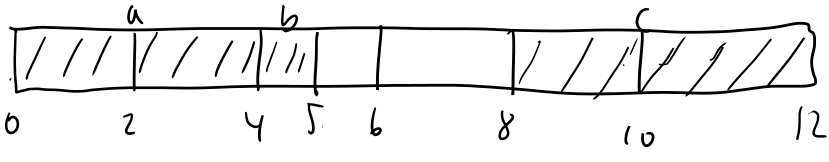
7. The processor can generate 2^{22} memory addresses which is 4,194,304 addresses. If each cell in the memory has a size of 16 bits, then it would hold 16 x 4194304 bits which is 8,388,608 bytes.

8. a = 4 bytes, b = 1 byte, c = 4 bytes



∴ the compiler would have to allocate 10 bytes

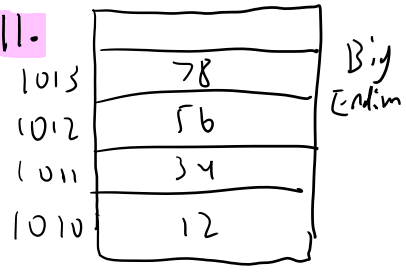
9.

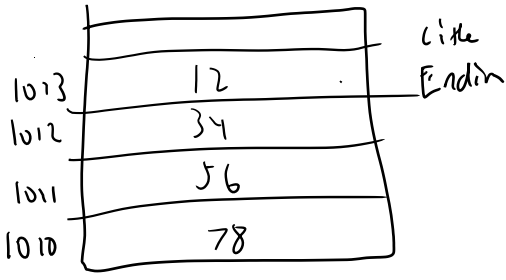


∴ the compiler has to allocate 12 bytes

10. We would get an error because words and longwords must be at even memory addresses.

11.





∴ For Big Endian, the memory address of the byte $34 would be $1011
For Little Endian, the memory address of the byte $34 would be $1012

12. After getting an output of 78 56 34 12, we know that the code ran on a small Endian byte ordering as the original hex was 0x12345678.

13. MOVE.B #$46, D0

C = 0 Z = 0
V = 0 N = 0
} since positive non 0 number

SUB.B #$51, D0 ($46 − $51)

46_{16}: 01000110_2 (same as unsigned)

-51_{10}: 01010001_2 → flip → 10101110 → add 1 → 10101111

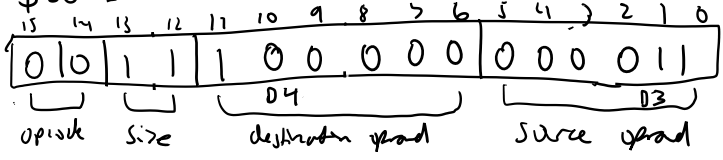
$$\begin{array}{r} 10101111 \\ +\ 01000110 \\ \hline 11110101 \end{array}$$

C = 1 Z = 0
V = 0 N = 1

we flip this for subtraction

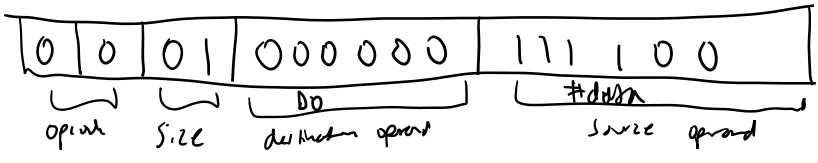
∴ This means the result is negative with a carry bit

14. $3803 = 0011100000000011



∴ MOVE.W D3, D4

15. $103C = 0001000000111100



∴ MOVE.B #$9, D0