Jose Capestany

Sayaf Almeri

EE215

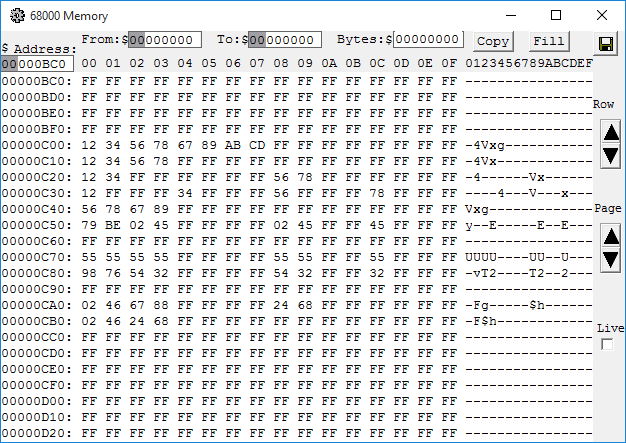
1/27/2016

LAB1 Report

The purpose of this lab was to get familiar with programming in assembly. The lab required us to use basic assembly directives in order to do basic computation such as addition or subtraction. It also teaches us how memory is stored, used, and called upon in order to provide these basic functions

We were first asked to define X and Y in order to use them later in the program. We used the DC.L assembly directive to define a 32-bit constant for X and Y. Then we had to define X0-X3 and Y0-Y3. We defined them as X0 is X + 1 and X1 is X + 2 and so forth. This is not adding a number to the constant, but instead is defining the memory value. Since X is stored in M[$C00] then X1 is the information stored in M[$C00] + 1, also known as M[$C01]. Once those were defined, we used them to carry out the next 20 operations. We will condense what occurred in these operations in the following lines:

1. Copy 32-bits to M[$C10] from M[X0]
2. Copy 16-bits to M[$C20] from M[X0]
3. Copy 16-bits to M[$C28] from M[X2]
4. Copy 8-bits to M[$C30] from M[X0]
5. Copy 8-bits to M[$C34] from M[X1]
6. Copy 8-bits to M[$C38] from M[X2]
7. Copy 8-bits to M[$C3C] from M[X3]
8. Copy 32-bits to M[$C40] from M[X2]
9. Copy 32-bits to D0 from M[X0] and then add 32-bits to D0 from M[Y0]. Copy 32-bits of D0 into M[$C50].
10. Copy 16-bits to D0 from M[X2] and then add 16-bits to D0 from M[Y2]. Copy 16-bits of D0 into M[$C58].
11. Copy 8-bits to D0 from M[X3] and then add 8-bits to D0 from M[Y3]. Copy 8-bits of D0 to M[$C5C].
12. Copy 32-bits to D0 from M[Y0] and then subtract 32-bits from D0 from M[X0]. Copy 32-bits of D0 into M[$C70].
13. Copy 16-bits to D0 from M[Y2] and then subtract 16-bits from D0 from M[X2]. Copy 16-bits of D0 into M[$C78].
14. Copy 8-bits to D0 from M[Y3] and then subtract 8-bits from D0 from M[X3]. Copy 8-bits of D0 into M[$C7C].
15. Copy 32-bits to D0 from M[Y0]. Take the complement of D0 and store it in D0. Copy 32-bits of D0 into M[$C80].
16. Copy 16-bits to D0 from M[Y2]. Take the complement of D0 and store it in D0. Copy 16-bits of D0 into M[$C88].
17. Copy 8-bits to D0 from M[Y3]. Take the complement of D0 and store it in D0. Copy 8-bits of D0 into M[$C8C].
18. Copy 32-bits to D0 from M[X2] and then add 32-bits from M[Y2]. Copy 32-bits of D0 into M[$CA0].
19. Copy 16-bits to D0 from M[X0] and then add 16-bits from M[X0]. Copy 16-bits of D0 into M[$CA8].
20. Copy 32-bits of D0 *(which is still the same as in part s)* into M[$CB0].



*The results of LAB1.*

We were able to successfully complete this lab after a few minor hiccups. The results that came from this lab were relatively simple. Adding two numbers together should produce a number which is the sum of those two numbers. The minor hiccups mainly came from inexperience. We originally had a problem with defining X0-X3 and Y0-Y3. We thought we had to add 1 to the value stored in M[$C00] instead of making adding 1 to the memory location. Something that didn’t affect our results was we originally cleared D0 after every operation. We didn’t realize until later that when you copy something in D0 it replaces what was already there. After completing this lab we both came the conclusion that we are able to now program basic arithmetic in assembly.