Optimizing 65xxx Assembly Language Code

By Albert Neuburger

1991 A2-Central Summer Conference

1. Use ALL registers.

2. Use, whenever possible, values that are already in the registers.

3. Use processor flags to their fullest extent.

4. Avoid using the stack to temporarily hold values (unless your trying to save memory space), it's faster to do:

```
lda  adrs (or DP)
sta  adrs (or DP)
```

then to push on and pop off values from the stack.

5. Make effective use of Direct Page addresses for things like storing and retrieving values (it's faster then using absolute addressing)

6. Avoid, whenever possible, doing RMW (Read, Modify, Write) operations.

```
ex.  asl  adrs (or DP)
ex.  inc  adrs (or DP)
```

7. To avoid RMW operations use the registers, especially the index registers (registers x and y), effectively. Do this by holding often used values, which would normally be modified by an RMW operation, in the index registers. Why?

because it's faster to do something like:

```
txa
asl  (immediate addressing)
tax
```

which takes 6 cycles,

then to do:

```
asl  adrs (or DP)
```

which, under the best conditions, still takes 7 cycles.

8. If appropriate, use the 'bit' instruction to non-destructively check the setting of bits in the accumulator.

9. Keep in mind, one of the fastest ways to check whether a value in a register is negative or zero is to simply transfer (ex. tax, txa, tay, tya, txy, tyx) it to another register. (this is assuming you have a free register to transfer to)

10. Also keep in mind:

```
        jmp  adrs (3 cycles)
```

is 1 cycle faster then,

```
        brl  displacement (4 cycles)
```

but is not relocatable like 'brl', therefore (obviously) use 'jmp adrs' only in code which you know won't/can't be relocated.

11. Try to put repetitive code in areas where most of the routines using the code converge, this has the potential to save lots of space in an application.

12. (an extension of 11) For conditional branches, put code which will appear after the branch, whether it's taken or not, before the actual branch instruction, but ONLY if the code doesn't effect the processor flag being used for the conditional branch.

The not-so-nice way:

```
                    lda  adrs
                    sec
                    sbc  number
                    bne  branch
                    sta  adrs
                    ...
                    ...

              branch  sta  adrs
                      ...
                      ...
```

The incredibly-wonderful-stupendous way:

```
                                    lda  adrs
                                    sec
                                    sbc  number
                                    sta  adrs
                                    bne  branch
                                    ...
                                    ...

                                branch  ...
                                        ...
```

13. Make large pieces of repetitive code into subroutines, NOT into macros.

14. OFFICIAL DISCLAIMER: I personally do not recommend using the following information, but include it because I feel one should try to know as much about programming ones machine as possible.

In desperate cases, where one is NOT using the Stack and/or the direct page, the stack pointer and/or the DP register can be used as an additional register. Their main use would probably be for avoiding RMW operations (as discussed in 6 and 7). But there are a number of significant drawbacks to using them:

a. interrupts must be off.
b. the original setting(s) for the register(s) must be saved so they can be restored later on.
c. absolutely NO code must use the stack and/or direct page since you have no idea where it may be located from one point in your code to the next.
d. you lose some very powerful useful addressing modes

FINAL POINT: When to save space and when to go for speed...

Very often a person can take one approach to a problem and save a few bytes in memory or go with another which is slightly longer but runs a few cycles faster.

There is no rule to help decide when to use what approach. Generally, the particular application and some common sense should help one with that decision....So always plan things out, don't just start writing a 50k program (or any other for that matter) the moment it comes to you...you'll probably end up with a giant kludge, tons of bugs, and a massive headache, other then that you should be all right....Have Fun!!!