ECS PROJECT #06 – Extra Credit (rev 1)
CSCI-410 Spring 2013

There are a number of useful extensions that a more feature-rich compiler might offer. If your assembler implements one or more of these features, you can earn Extra Credit points. Since you are now allowing non-portable code, meaning code will assemble that will not assemble in other assemblers, the user must explicitly enable the extended features by appending a "-x" switch on the command line following the name of the input file. Without this switch, the assembler should be "strictly compliant" and only assemble code that the authors' assembler will accept.

The first extension is to permit the programmer to put the three possible targets of an assignment in any order, instead of the "**`AMD`**" order of the strictly compliant assembler.

The second extension is to support a generic compute instruction that allows the programmer to directly specify the seven ALU control signals. This instruction is of the form **`Xhh`** where '**`X`**' is a literal uppercase character and '**`hh`**' is the two digit hexademical value representing the bit pattern for the control signals. The bits are maped as follows:

bit	mapping
b0	c6 – no
b1	c5 – f
b2	c4 – ny
b3	c3 – zy
b4	c2 – nx
b5	c1 – zx
b6	a – a
b7	Reserved – must be 0

Thus **`X50`** = (!D)&(M).

The third extension is to permit the user to enter numeric constants in hexadecimal notation using the "0x" prefix. Hex characters should be case insensitive.

The fourth extension is to permit the programmer to define the value associated with symbols explicitly by following the "(symbol_name)" pseudoinstruction with the desired value on the same line. Thus, if the person wanted to associated the symbol "space" with the value 32, or alternatively 0x20, then they could write:

```
(space) 0x20
```

If no value is supplied, then the normal behavior should result.