Sim12 is based on the M68HC12AEVB. It assumes the following memory map:

- 16K External RAM at $4000-$7FFF
- 32K External ROM at $8000-$FFFF
- Internal RAM at $800-$BFF
- Internal EEPROM at $1000-$1FFF

Note: You can load in programs anywhere in the above mentioned areas but should only write to RAM (i.e. assume ROM and EEPROM are read only).

Assemble using the "MiniIDE" 6812 Assembler found on our web site. Inside of MiniIDE go to "Build" => "Options" and make sure the "Generate Listing File" option is checked.

---

Sim12 is based on the M68HC12AEVB. It assumes the following memory map:

- 16K External RAM at $4000-$7FFF
- 32K External ROM at $8000-$FFFF
- Internal RAM at $800-$BFF
- Internal EEPROM at $1000-$1FFF

Note: You can load in programs anywhere in the above mentioned areas but should only write to RAM (i.e. assume ROM and EEPROM are read only).

Assemble using the "MiniIDE" 6812 Assembler found on our web site. Inside of MiniIDE go to "Build" => "Options" and make sure the "Generate Listing File" option is checked.

---

```asm
* ASSEMBLER EQUATE (Define) Statements
CONST EQU $8000  ;starting address for constants in memory (Ext. ROM)
RAM1 EQU $4000  ;starting address for variables in memory (Ext. RAM)
RAM2 EQU $5000  ;starting address for additional variables in memory (Ext. RAM)

* DATA ALLOCATION SECTION - constants & variables
* Data can go before or after your program but should not in the middle of a program for clarity reasons.
* Special Note: Assembler directives are used to place data and variables into memory. They are not 6812 instructions and thus are not executed at run time. When this program is loaded into memory, the data (created above by the assembler) is also copied down into memory.

** Data Transfer Examples **
* Program #1 - moving a byte to a new location
  org      $1000  ;we will fill in comments in class
  ldaa     num1  ;
  staa     results ;
  end1  bra  end1
* Program #2 - moving a word (two bytes) to a new location
  org      $1100
  ldd      num2  ;
```

---

Sim12 is based on the M68HC12AEVB. It assumes the following memory map:

- 16K External RAM at $4000-$7FFF
- 32K External ROM at $8000-$FFFF
- Internal RAM at $800-$BFF
- Internal EEPROM at $1000-$1FFF

Note: You can load in programs anywhere in the above mentioned areas but should only write to RAM (i.e. assume ROM and EEPROM are read only).

Assemble using the "MiniIDE" 6812 Assembler found on our web site. Inside of MiniIDE go to "Build" => "Options" and make sure the "Generate Listing File" option is checked.

---

```asm
* ASSEMBLER EQUATE (Define) Statements
CONST EQU $8000  ;starting address for constants in memory (Ext. ROM)
RAM1 EQU $4000  ;starting address for variables in memory (Ext. RAM)
RAM2 EQU $5000  ;starting address for additional variables in memory (Ext. RAM)

* DATA ALLOCATION SECTION - constants & variables
* Data can go before or after your program but should not in the middle of a program for clarity reasons.
* Special Note: Assembler directives are used to place data and variables into memory. They are not 6812 instructions and thus are not executed at run time. When this program is loaded into memory, the data (created above by the assembler) is also copied down into memory.

** Data Transfer Examples **
* Program #1 - moving a byte to a new location
  org      $1000  ;we will fill in comments in class
  ldaa     num1  ;
  staa     results ;
  end1  bra  end1
* Program #2 - moving a word (two bytes) to a new location
  org      $1100
  ldd      num2  ;
```

---

Sim12 is based on the M68HC12AEVB. It assumes the following memory map:

- 16K External RAM at $4000-$7FFF
- 32K External ROM at $8000-$FFFF
- Internal RAM at $800-$BFF
- Internal EEPROM at $1000-$1FFF

Note: You can load in programs anywhere in the above mentioned areas but should only write to RAM (i.e. assume ROM and EEPROM are read only).

Assemble using the "MiniIDE" 6812 Assembler found on our web site. Inside of MiniIDE go to "Build" => "Options" and make sure the "Generate Listing File" option is checked.

---

```asm
* ASSEMBLER EQUATE (Define) Statements
CONST EQU $8000  ;starting address for constants in memory (Ext. ROM)
RAM1 EQU $4000  ;starting address for variables in memory (Ext. RAM)
RAM2 EQU $5000  ;starting address for additional variables in memory (Ext. RAM)

* DATA ALLOCATION SECTION - constants & variables
* Data can go before or after your program but should not in the middle of a program for clarity reasons.
* Special Note: Assembler directives are used to place data and variables into memory. They are not 6812 instructions and thus are not executed at run time. When this program is loaded into memory, the data (created above by the assembler) is also copied down into memory.

** Data Transfer Examples **
* Program #1 - moving a byte to a new location
  org      $1000  ;we will fill in comments in class
  ldaa     num1  ;
  staa     results ;
  end1  bra  end1
* Program #2 - moving a word (two bytes) to a new location
  org      $1100
  ldd      num2  ;
```
* Program #3 - load immediate data, transfer to another reg & store in mem
  org  $1200
  ldaa  #%10101110 ;
  tab ;
  stab  results ;
  end3
  bra  end3

* Program #4 - move data using an index register
  org  $1300
  ldx  #num1  ;
  ldy  #results ;
  ldab  0,x  ;
  staa  2,y  ;
  end4
  bra  end4

* ARITHMETIC & LOGIC EXAMPLES
* Program #5 - adding (2) byte size numbers & saving the result
  org  $1400
  ldab  value1  ;
  aba ;
  staa  sum
  end5
  bra  end5

* Program #6 - subtracting a byte in memory from an immediate data byte
  org  $1500
  ldab  #$20  ;
  subb  value2  ;
  ldx  #sum  ;
  stab  0,x
  end6
  bra  end6

* Program #7 - summing words
  org  $1600
  ldd  num2  ;
  addd  num2  ;
  ldy  #results ;
  std  9,y  ;
  end7
  bra  end7

* Program #8 - bit wise ANDing example
  org  $1700
  ldab  num1  ;
  nega  ;
  andb  value1  ;
  staa  sum
  end8
  bra  end8