The memory maps

<table>
<thead>
<tr>
<th></th>
<th>9S12C32</th>
<th>MC68HC812A4</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O</td>
<td>$0000$ to $03FF$</td>
<td>$0000$ to $01FF$</td>
</tr>
<tr>
<td>Ports</td>
<td>AD, M, S, T</td>
<td>A,B,C,D,E,F,H,J,S,T,AD</td>
</tr>
<tr>
<td>RAM</td>
<td>$3800$ to $3FFF$</td>
<td>$0800$ to $0BFF$</td>
</tr>
<tr>
<td></td>
<td>2K</td>
<td>1K</td>
</tr>
<tr>
<td>ROM</td>
<td>$4000$ to $7FFF$</td>
<td>$F000$ to $FFFF$</td>
</tr>
<tr>
<td></td>
<td>$8000$ to $FFFF$</td>
<td>4K</td>
</tr>
</tbody>
</table>

Review: simplified execution has five phases:

<table>
<thead>
<tr>
<th>Phase</th>
<th>Function</th>
<th>R/W</th>
<th>Address</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Op code fetch</td>
<td>read</td>
<td>PC++</td>
<td>Put data in IR</td>
</tr>
<tr>
<td>2</td>
<td>Operand fetch</td>
<td>read</td>
<td>PC++</td>
<td>Calculate EA</td>
</tr>
<tr>
<td>3</td>
<td>Data read</td>
<td>read</td>
<td>SP, EAR</td>
<td>Data (ALU)</td>
</tr>
<tr>
<td>4</td>
<td>Free cycle</td>
<td>read</td>
<td>PC/SP/$FFFF$</td>
<td>ALU (sets CCR)</td>
</tr>
<tr>
<td>5</td>
<td>Data store</td>
<td>write</td>
<td>SP, EAR</td>
<td>Results stored</td>
</tr>
</tbody>
</table>

4. **Extended addressing** mode (review)
   - uses a 16-bit address
   - size of data depends on the op code (which register is uses)

Addressing Modes, Machine Code, and Execution Times:

<table>
<thead>
<tr>
<th>Source Form</th>
<th>Address Mode</th>
<th>Object Code</th>
<th>Cycles</th>
<th>Access Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA opr16a</td>
<td>EXT</td>
<td>B6 hh 11</td>
<td>3</td>
<td>rOP</td>
</tr>
<tr>
<td>LDX opr16a</td>
<td>EXT</td>
<td>FE hh 11</td>
<td>3</td>
<td>rOP</td>
</tr>
<tr>
<td>STX opr16a</td>
<td>EXT</td>
<td>7E hh 11</td>
<td>3</td>
<td>rOPW</td>
</tr>
<tr>
<td>INC opr16a</td>
<td>EXT</td>
<td>72 hh 11</td>
<td>4</td>
<td>rOPw</td>
</tr>
</tbody>
</table>

**How to do Lab 3.2**
1) look at TExaS assembly listing
2) Work through the phases
3) Verify by running the TExaS simulation

Assume this initial state
PC = $F02F
$0800 = $0D
$0801 = $0A

1) look at TExaS assembly listing
$F02F FE0800   [3](45){ROP} ldx $0800

2) Work through the phases

<table>
<thead>
<tr>
<th>R/W</th>
<th>Addr</th>
<th>Data</th>
<th>changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>$F02F</td>
<td>$FE</td>
<td>PC=$F030, IR=$FE</td>
</tr>
<tr>
<td>R</td>
<td>$F030</td>
<td>$08</td>
<td>PC=$F031</td>
</tr>
<tr>
<td>R</td>
<td>$F031</td>
<td>$00</td>
<td>PC=$F032, EAR=$0800</td>
</tr>
<tr>
<td>R</td>
<td>$0800</td>
<td>$0D</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>$0801</td>
<td>$0A</td>
<td>X=$0D0A</td>
</tr>
</tbody>
</table>

3) TExaS simulation shows the following cycles.
Opcode fetch   R 0xF02F 0xFE from EEPROM
Operand fetch   R 0xF030 0x08 from EEPROM
Operand fetch   R 0xF031 0x00 from EEPROM
Fetch msb @ EARR 0x0800 0xD from RAM
Fetch lsb @ EARR 0x0801 0xA from RAM
0xF02F ldx $0800
A=$0F B=$0A CC=sXhInzvc PC=$F032 X=$0D0A

1) look at TExaS assembly listing
$F032 7E0802 [3][48]{WOP} stx $0802

2) Work through the phases

<table>
<thead>
<tr>
<th>R/W</th>
<th>Addr</th>
<th>Data</th>
<th>changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>$F032</td>
<td>$7E</td>
<td>PC=$F033, IR=$7E</td>
</tr>
<tr>
<td>R</td>
<td>$F033</td>
<td>$08</td>
<td>PC=$F034</td>
</tr>
<tr>
<td>R</td>
<td>$F034</td>
<td>$02</td>
<td>PC=$F035, EAR=$0802</td>
</tr>
<tr>
<td>W</td>
<td>$0802</td>
<td>$0D</td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>$0803</td>
<td>$0A</td>
<td>($0802)=$0D0A</td>
</tr>
</tbody>
</table>

3) TExaS simulation shows the following cycles.
Opcode fetch   R 0xF032 0x7E from EEPROM
Operand fetch  R 0xF033 0x08 from EEPROM
Operand fetch  R 0xF034 0x02 from EEPROM
Store msb @ EARW 0x0802 0xD to RAM
Store lsb @ EARW 0x0803 0xA to RAM
0xF032 stx $0802
A=$0F B=$0A CC=sXhInzvc PC=$F035 X=$0D0A
1) look at TExaS assembly listing
$F035 720800 \{rOPw \} \text{inc} $0800

2) Work through the phases

<table>
<thead>
<tr>
<th>R/W</th>
<th>Addr</th>
<th>Data</th>
<th>changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>$F035</td>
<td>$72</td>
<td>PC=$F036, IR=$72</td>
</tr>
<tr>
<td>R</td>
<td>$F036</td>
<td>$08</td>
<td>PC=$F037</td>
</tr>
<tr>
<td>R</td>
<td>$F037</td>
<td>$00</td>
<td>PC=$F038, EAR=$0800</td>
</tr>
<tr>
<td>R</td>
<td>$0800</td>
<td>$0D</td>
<td>[0800]=$0E</td>
</tr>
<tr>
<td>W</td>
<td>$0800</td>
<td>$0E</td>
<td>[0800]=$0E</td>
</tr>
</tbody>
</table>

3) TExaS simulation shows the following cycles.

- Opcode fetch: R 0xF035 0x72 from EEPROM
- Operand fetch: R 0xF036 0x08 from EEPROM
- Operand fetch: R 0xF037 0x00 from EEPROM
- Fetch using EARR: 0x0800 0x0D from RAM
- Store using EARW: 0x0800 0x0E to RAM

0xF035 inc $0800
A=$0F B=$0A CC=sXhInzvc PC=$F038 X=$0D0A

5. PC Relative addressing mode
- used for the branch instructions
- stored in the machine code is not the absolute address
- but the 8-bit signed offset relative distance from the current PC value
- the PC already points to the next instruction
- the assembler calculates it for us
**Example 1. Backward branch**

1) address of current instruction

\( \$F880 \)  \( \text{bra} \)  \( \$F840 \)

2) op code of branch instruction, and size of instruction

\( \$F880 \)  \( \$20rr \)  \( \text{bra} \)  \( \$F840 \)

3) address of next instruction

\( \$F882 \)

4) calculate PC relative addressing

\[ rr = \text{destination} - \text{address of next instruction} \]

\[ = \$F840 - \$F882 = -$42 = $BE \]

The object code for this instruction will be

\( \$F880 \)  \( \$20BE \)  \( \text{bra} \)  \( \$F840 \)

**Example 2. Forward branch**

1) address of current instruction

\( \$F000 \)  \( \text{bra} \)  \( \$F044 \)

2) op code of branch instruction, and size of instruction

\( \$F000 \)  \( \$20rr \)  \( \text{bra} \)  \( \$F044 \)

3) address of next instruction

\( \$F000 \)  \( \$20rr \)  \( \text{bra} \)  \( \$F044 \)
4) calculate PC relative addressing  
\[ rr = \text{destination} - \text{address of next instruction} \]
\[ = \$F044 - \$F002 = \$42 \]
The object code for this instruction will be
\[ \$F000 \quad \$2042 \quad \text{bra} \quad \$F044 \]

**Example 2. Branch out of range error**

1) address of current instruction
\[ \$F000 \quad \text{bra} \quad \$F144 \]
2) op code of branch instruction, and size of instruction
\[ \$F000 \quad \$20rr \quad \text{bra} \quad \$F144 \]
3) address of next instruction
\[ \$F000 \quad \$20rr \quad \text{bra} \quad \$F144 \]
\[ \$F002 \]
4) calculate PC relative addressing  
\[ rr = \text{destination} - \text{address of next instruction} \]
\[ = \$F144 - \$F002 = \$142 \]
“Branch out of range” assembly error.

### 3.5.5. Additional addressing modes on the 6812

1. **Indexed addressing mode**
   - fixed offset with the 16-bit registers: X, Y, SP, or PC
   - 5-bit (-16 to +15),
   - 9-bit (-256 to +255), or
   - 16-bit unsigned (0 to 65535) or signed (-32768 to +32767.)

**Case 1: Five bit (-16 to +15) index mode**

<table>
<thead>
<tr>
<th>machine code</th>
<th>opcode</th>
<th>operand</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>$6A5C</td>
<td>staa</td>
<td>-4,Y</td>
<td>[Y-4] = RegA</td>
</tr>
</tbody>
</table>

Assuming Register Y=$0823, leave Register Y unchanged.

\[
\begin{array}{c|c|c|c}
Y & \$0823 & \$081E & \$F800 \\
A & \$56   & \$0820 & \$F801 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c|c}
\text{RAM} & \text{EEPROM} & \text{staa} & -4,Y \\
\$0821 & \$56 & \$F802 & \$5C \\
\end{array}
\]

*Figure 3.27. Example of the 6812 indexed addressing mode.*
1) look at TExaS assembly listing
$F801 6A5C [ 2]( 3){Pw } staa -4,Y

2) Work through the phases

<table>
<thead>
<tr>
<th>R/W</th>
<th>Addr</th>
<th>Data</th>
<th>changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>$F801</td>
<td>$6A</td>
<td>PC=$F802, IR=$6A</td>
</tr>
<tr>
<td>R</td>
<td>$F802</td>
<td>$5C</td>
<td>PC=$F803, EAR=$081F</td>
</tr>
<tr>
<td>W</td>
<td>$081F</td>
<td>$56</td>
<td>[$081F]=$56</td>
</tr>
</tbody>
</table>

3) TExaS simulation shows the following cycles.
 Opcode fetch R 0xF801 0x6A from EEPROM
 Operand fetch R 0xF802 0x5C from EEPROM
 Store using EARW 0x081F 0x56 to RAM
 0xF801 staa -4,Y
 PC=$F803 RegA=$56 RegY=$0823

Case 2: Nine bit (-256 to +255) indexed mode

<table>
<thead>
<tr>
<th>machine code</th>
<th>opcode</th>
<th>operand</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>$6AE840</td>
<td>staa</td>
<td>$40,Y</td>
<td>[Y+$40] = RegA</td>
</tr>
</tbody>
</table>

Assuming Register Y=$0823, leaves Register Y unchanged.

Figure 3.28. Another example of the 6812 indexed addressing mode.
1) look at TExaS assembly listing
$F801 6AE828 [3](3)\{PwO\} staa 40,Y

2) Work through the phases

<table>
<thead>
<tr>
<th>R/W</th>
<th>Addr</th>
<th>Data</th>
<th>changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>$F801</td>
<td>$6A</td>
<td>PC=$F802, IR=$6A</td>
</tr>
<tr>
<td>R</td>
<td>$F802</td>
<td>$E8</td>
<td>PC=$F803</td>
</tr>
<tr>
<td>R</td>
<td>$F803</td>
<td>$28</td>
<td>PC=$F804, EAR=$084B</td>
</tr>
<tr>
<td>W</td>
<td>$084B</td>
<td>$56</td>
<td>[$084B]=$56</td>
</tr>
</tbody>
</table>

3) TExaS simulation shows the following cycles.

 Opcode fetch  R 0xF801 0x6A from EEPROM
 Operand fetch R 0xF802 0xE8 from EEPROM
 Operand fetch R 0xF803 0x28 from EEPROM
 Store using EARW 0x084B 0x56 to RAM
 0xF801 staa 40,Y
 PC=$F804 RegA=$56 RegY=$0823

Case 3: Sixteen bit indexed mode

<table>
<thead>
<tr>
<th>machine code</th>
<th>opcode</th>
<th>Operand</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>$6AE0200</td>
<td>staa</td>
<td>$200,Y</td>
<td>[Y+$200] = RegA</td>
</tr>
</tbody>
</table>

Assuming Register Y=$0823, leaves Register Y unchanged.

Figure 3.29. A third example of the 6812 indexed addressing mode.
1) look at TExaS assembly listing

$F801 6AEA0200 [ 3]( 3){PwP} staa $200,y

2) Work through the phases

<table>
<thead>
<tr>
<th>R/W</th>
<th>Addr</th>
<th>Data changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>$F801</td>
<td>$6A</td>
</tr>
<tr>
<td></td>
<td>PC=$F802, IR=$6A</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>$F802</td>
<td>$EA</td>
</tr>
<tr>
<td></td>
<td>PC=$F803</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>$F803</td>
<td>$02</td>
</tr>
<tr>
<td></td>
<td>PC=$F804</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>$F804</td>
<td>$00</td>
</tr>
<tr>
<td></td>
<td>PC=$F805, EAR=$0A23</td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>$0A23</td>
<td>$56</td>
</tr>
<tr>
<td></td>
<td>[$0A23]=$56</td>
<td></td>
</tr>
</tbody>
</table>

3) TExaS simulation shows the following cycles.

Opcode fetch R 0xF801 0x6A from EEPROM
Operand fetch R 0xF802 0xEA from EEPROM
Operand fetch R 0xF803 0x02 from EEPROM
Operand fetch R 0xF804 0x00 from EEPROM
Store using EARW 0x0A23 0x56 to RAM
0xF801 staa 512,Y
PC=$F805 RegA=$56 RegY=$0823

Properties of 16-bit addition

• offset can be unsigned (0 to 65535) or
• signed (-32768 to +32767.)

Indexed mode is useful

• data structures
• parameters on the stack.
• local variables on the stack.
Lab 3.2: Start a fresh copy of TExaS and create a new Program and Microcomputer windows. Execute the Mode->Processor… command and select the either the MC68HC711E9 or the MC68HC812A4. Save these two documents with names having the same root name. E.g., Lab3.rtf Lab3.uc. Click on the Program window and execute the Assemble->Options… command. Make sure the check boxes make the configuration shown in Figure 3.36.

Step 1. Type the assembly source code from Program 3.17. Assemble the program by executing the Assemble->Assemble command.

Step 2. Execute this program by hand up to an including the stop instruction. For each instruction show the memory cycles generated and the values of any registers than change. For the 6811, show the actual cycles. For the 6812, show the simplified cycles as described in section 3.2.5.
Step 3. Activate the **FollowPC CycleView InstructionView** and **LogRecord** modes using the commands in the Mode menu. Single step the program up to an including the stop instruction. Verify the answers you gave for Step 2.

```assembly
; MC68HC812A4
org $F000
ldx #$0800
ldy #$0900
ldd #$1234
ldaa $50,x
ldab $1A,y
ldaa $AC,x
ldaa -13,y
ldd $EF,x
ldd -$0B,y
```
stop
org $FFFE
fdb $F000