This is a closed book exam. You must put your answers on this piece of paper only. You have 50 minutes, so allocate your time accordingly. Please read the entire quiz before starting.

(5) Question 1.

(5) Question 2.

(5) Question 3.

(5) Question 4.

(5) Question 5.

(5) Question 6.

(10) Question 7.

<table>
<thead>
<tr>
<th>R/W</th>
<th>Addr</th>
<th>Data</th>
<th>Changes to A, B, X, Y, S, PC, IR, EAR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(20) Question 8.

(20) Question 9.

(20) Question 10.
orcc 8-bit logical or to RegCC
psah push 8-bit RegA onto stack
psbh push 8-bit RegB onto stack
pshc push 8-bit RegCC onto stack
psdh push 16-bit RegD onto stack
pshx push 16-bit RegX onto stack
pshy push 16-bit RegY onto stack
pula pop 8 bits off stack into RegA
pulb pop 8 bits off stack into RegB
pulc pop 8 bits off stack into RegCC
puld pop 16 bits off stack into RegD
pulx pop 16 bits off stack into RegX
puly pop 16 bits off stack into RegY
rev Fuzzy logic rule evaluation
revw weighted Fuzzy rule evaluation
rol 8-bit roll shift left Memory
rola 8-bit roll shift left RegA
rolb 8-bit roll shift left RegB
ror 8-bit roll shift right Memory
rora 8-bit roll shift right RegA
rorb 8-bit roll shift right RegB
rtc return sub in expanded memory
rti return from interrupt
rts return from subroutine
sba 8-bit subtract RegA-RegB
sbca 8-bit sub with carry from RegA
sbcb 8-bit sub with carry from RegB
sec set carry bit, C=1
sei set I=1, disable interrupts
sev set overflow bit, V=1
sex sign extend 8-bit to 16-bit reg
staa 8-bit store memory from RegA
stab 8-bit store memory from RegB
std 16-bit store memory from RegD
sts 16-bit store memory from SP
stx 16-bit store memory from RegX
sty 16-bit store memory from RegY
suba 8-bit sub from RegA
subb 8-bit sub from RegB
subd 16-bit sub from RegD
swi software interrupt, trap
tabl 8-bit look up and interpolation
tbne test and branch if result=0
tbeq test and branch if result=0
rev Fuzzy logic rule evaluation
revw weighted Fuzzy rule evaluation
rol 8-bit roll shift left Memory
rola 8-bit roll shift left RegA
rolb 8-bit roll shift left RegB
ror 8-bit roll shift right Memory
rora 8-bit roll shift right RegA
rorb 8-bit roll shift right RegB
rtc return sub in expanded memory
rti return from interrupt
rts return from subroutine
sba 8-bit subtract RegA-RegB
sbca 8-bit sub with carry from RegA
sbcb 8-bit sub with carry from RegB
sec set carry bit, C=1
sei set I=1, disable interrupts
sev set overflow bit, V=1
sex sign extend 8-bit to 16-bit reg
staa 8-bit store memory from RegA
stab 8-bit store memory from RegB
std 16-bit store memory from RegD
sts 16-bit store memory from SP
stx 16-bit store memory from RegX
sty 16-bit store memory from RegY
suba 8-bit sub from RegA
subb 8-bit sub from RegB
subd 16-bit sub from RegD
swi software interrupt, trap
tabl 8-bit look up and interpolation
tbne test and branch if result=0
tbeq test and branch if result=0
rev Fuzzy logic rule evaluation
revw weighted Fuzzy rule evaluation
rol 8-bit roll shift left Memory
rola 8-bit roll shift left RegA
rolb 8-bit roll shift left RegB
ror 8-bit roll shift right Memory
rora 8-bit roll shift right RegA
rorb 8-bit roll shift right RegB
rtc return sub in expanded memory
rti return from interrupt
rts return from subroutine
sba 8-bit subtract RegA-RegB
sbca 8-bit sub with carry from RegA
sbcb 8-bit sub with carry from RegB
sec set carry bit, C=1
sei set I=1, disable interrupts
sev set overflow bit, V=1
sex sign extend 8-bit to 16-bit reg
staa 8-bit store memory from RegA
stab 8-bit store memory from RegB
std 16-bit store memory from RegD
sts 16-bit store memory from SP
stx 16-bit store memory from RegX
sty 16-bit store memory from RegY
suba 8-bit sub from RegA
subb 8-bit sub from RegB
subd 16-bit sub from RegD
swi software interrupt, trap
tabl 8-bit look up and interpolation
tbne test and branch if result=0
tbeq test and branch if result=0
rev Fuzzy logic rule evaluation
revw weighted Fuzzy rule evaluation
rol 8-bit roll shift left Memory
rola 8-bit roll shift left RegA
rolb 8-bit roll shift left RegB
ror 8-bit roll shift right Memory
rora 8-bit roll shift right RegA
rorb 8-bit roll shift right RegB
rtc return sub in expanded memory
rti return from interrupt
rts return from subroutine
sba 8-bit subtract RegA-RegB
sbca 8-bit sub with carry from RegA
sbcb 8-bit sub with carry from RegB
sec set carry bit, C=1
sei set I=1, disable interrupts
sev set overflow bit, V=1
sex sign extend 8-bit to 16-bit reg
staa 8-bit store memory from RegA
stab 8-bit store memory from RegB
std 16-bit store memory from RegD
sts 16-bit store memory from SP
stx 16-bit store memory from RegX
sty 16-bit store memory from RegY
suba 8-bit sub from RegA
subb 8-bit sub from RegB
subd 16-bit sub from RegD
swi software interrupt, trap
tabl 8-bit look up and interpolation
tbne test and branch if result=0
tbeq test and branch if result=0
Freescale 6812 addressing modes \( \mathbf{r} \) is \( \mathbf{X}, \mathbf{Y}, \mathbf{SP}, \) or \( \mathbf{PC} \)

<table>
<thead>
<tr>
<th>example</th>
<th>addressing mode</th>
<th>Effective Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>ldaa #u</td>
<td>immediate</td>
<td>No EA</td>
</tr>
<tr>
<td>ldaa u</td>
<td>direct</td>
<td>EA is 8-bit address (0 to 255)</td>
</tr>
<tr>
<td>ldaa U</td>
<td>extended</td>
<td>EA is a 16-bit address</td>
</tr>
<tr>
<td>ldaa m,r</td>
<td>5-bit index</td>
<td>EA=r+m (-16 to 15)</td>
</tr>
<tr>
<td>ldaa v,+r</td>
<td>pre-increment</td>
<td>r=r+v, EA=r (1 to 8)</td>
</tr>
<tr>
<td>ldaa v,-r</td>
<td>pre-decrement</td>
<td>r=r-v, EA=r (1 to 8)</td>
</tr>
<tr>
<td>ldaa v,r+</td>
<td>post-increment</td>
<td>EA=r, r=r+v (1 to 8)</td>
</tr>
<tr>
<td>ldaa v,r-</td>
<td>post-decrement</td>
<td>EA=r, r=r-v (1 to 8)</td>
</tr>
<tr>
<td>ldaa A,r</td>
<td>Reg A offset</td>
<td>EA=r+A, zero padded</td>
</tr>
<tr>
<td>ldaa B,r</td>
<td>Reg B offset</td>
<td>EA=r+B, zero padded</td>
</tr>
<tr>
<td>ldaa D,r</td>
<td>Reg D offset</td>
<td>EA=r+D</td>
</tr>
<tr>
<td>ldaa q,r</td>
<td>9-bit index</td>
<td>EA=q (-256 to 255)</td>
</tr>
<tr>
<td>ldaa W,r</td>
<td>16-bit index</td>
<td>EA=W (-32768 to 65535)</td>
</tr>
<tr>
<td>ldaa [D,r]</td>
<td>D indirect</td>
<td>EA=(r+D)</td>
</tr>
<tr>
<td>ldaa [W,r]</td>
<td>indirect</td>
<td>EA=(r+W) (-32768 to 65535)</td>
</tr>
</tbody>
</table>

**Pseudo op meaning**

- org = equ Specific absolute address to put subsequent object code
- = equ Define a constant symbol
- set Define or redefine a constant symbol
- dc.b db fcb .byte Allocate byte(s) of storage with initialized values
- fcc Create an ASCII string (no termination character)
- dc.w dw fdb .word Allocate word(s) of storage with initialized values
- dc.l dl .long Allocate 8-bit word(s) of storage with initialized values
- ds ds.b rmb .blkb Allocate 32-bit long word(s) of storage with initialized values
- ds.w .blkw Allocate bytes of storage without initialization
- ds.l .blkl Allocate 32-bit words of storage without initialization

Jonathan W. Valvano  September 26, 2008  10:00am-10:50am
## SUBD Subtract Double Accumulator

**Operation:** \((A : B) - (M : M + 1) \rightarrow A : B\)

**Description:** Subtracts the content of memory location \(M : M + 1\) from the content of double accumulator \(D\) and places the result in \(D\).

<table>
<thead>
<tr>
<th>Source Form</th>
<th>Address Mode</th>
<th>Object Code</th>
<th>HCS12 Access Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBD #opr16i</td>
<td>IMM</td>
<td>83 jj kk</td>
<td>PO</td>
</tr>
<tr>
<td>SUBD opr8a</td>
<td>DIR</td>
<td>93 dd</td>
<td>Rpf</td>
</tr>
<tr>
<td>SUBD opr16a</td>
<td>EXT</td>
<td>B3 hh ll</td>
<td>RPO</td>
</tr>
<tr>
<td>SUBD oprx0_xyssp</td>
<td>IDX</td>
<td>A3 xb</td>
<td>Rpf</td>
</tr>
<tr>
<td>SUBD oprx9,xyssp</td>
<td>IDX1</td>
<td>A3 xb ff</td>
<td>RPO</td>
</tr>
<tr>
<td>SUBD oprx16,xyssp</td>
<td>IDX2</td>
<td>A3 xb ee ff</td>
<td>fRPP</td>
</tr>
</tbody>
</table>