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.** Give the decimal value……

(5) **Part 2a.** Specify 0 or 1 ............

(5) **Part 2b.** Specify 0 or 1 ............

(5) **Question 3.** Specify A-H ............

(5) **Question 4.** Give the range....

(5) **Question 5.** Show the machine code....

(5) **Question 6.** Give example inputs, specify “none” if none exist.

(5) **Question 7.** Give example inputs, specify “none” if none exist.

(5) **Part 8a.** What value is pushed?............

(10) **Part 8b.** Simplified memory cycles (you may or may not need all 5 entries)

<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>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
(15) **Question 9.** Write the assembly language. (not a subroutine or a main program, just instructions)

```
DDRM equ $0252  ; Port M Direction
DDRT equ $0242  ; Port T Direction
PTM  equ $0250  ; Port M I/O Register
PTT  equ $0240  ; Port T I/O Register
```

(30) **Question 10.** Write the assembly language subroutine. (Do not include the main program)

```
;********Set5************
; Initializes memory $3900 to 397F to value 5
; inputs: none
; outputs: none
; errors: none
Set5
```

```
  rts
```
aba  8-bit add RegA+RegB
abx  unsigned add RegX+RegB
aby  unsigned add RegY+RegB
adca 8-bit add with carry to RegA
adcb 8-bit add with carry to RegB
add  8-bit add to RegA
addb 8-bit add to RegB
addh 16-bit add to RegD
anda 8-bit logical and to RegA
andb 8-bit logical and to RegB
anddb 8-bit logical and to RegB
andd 8-bit logical and to RegD
asld/lsld 8-bit left shift Memory
aslb/lslb 8-bit arith left shift RegB
asld/lsld16 8-bit left shift RegD
asr  8-bit arith right shift Memory
asra 8-bit arith right shift
asrb 8-bit arith right shift to RegB
bcc  branch if carry clear
bcrl clear bits in memory
bcs  branch if carry set
beq  branch if result is zero (Z=1)
bege branch if signed =
beqd enter background debug mode
bgt  branch if signed >
bhi  branch if unsigned >
bits 8-bit and with RegA, sets CCR
bitb 8-bit and with RegB, sets CCR
ble  branch if signed <=
bls  branch if unsigned <
bit  branch if signed <
bbmi branch if result is negative (N=1)
bbne branch if result is nonzero (Z=0)
bbpl branch if result is positive (N=0)
bra  branch always
brclir branch if bits are clear,
brn  branch never
brset branch if bits are set
bset set bits in memory
bsr  branch to subroutine
bvc  branch if overflow clear
bvs  branch if overflow set
call subroutine in expanded memory
clb  branch if signed =
clls  branch if signed =
clbf branch if signed <
clbs branch if signed <
clbi branch if signed <
blmi branch if result is negative
blne branch if result is nonzero
blpl branch if result is positive
bra  branch always
brccir branch if bits are clear,
bra  branch never
brset branch if bits are set
bset  set bits in memory
bsr  branch to subroutine
bvc  branch if overflow clear
bvs  branch if overflow set
call subroutine in expanded memory
clb  branch if signed =
clls  branch if signed =
clbf branch if signed <
clbs branch if signed <
clbi branch if signed <
blmi branch if result is negative
blne branch if result is nonzero
blpl branch if result is positive
bra  branch always
brccir branch if bits are clear,
bra  branch never
brset branch if bits are set
bset set bits in memory
bsr  branch to subroutine
bvc  branch if overflow clear
bvs  branch if overflow set
call subroutine in expanded memory
clb  branch if signed =
clls  branch if signed =
clbf branch if signed <
clbs branch if signed <
clbi branch if signed <
blmi branch if result is negative
blne branch if result is nonzero
blpl branch if result is positive
bra  branch always
brccir branch if bits are clear,
bra  branch never
brset branch if bits are set
bset set bits in memory
bsr  branch to subroutine
bvc  branch if overflow clear
bvs  branch if overflow set
call subroutine in expanded memory
clb  branch if signed =
clls  branch if signed =
clbf branch if signed <
clbs branch if signed <
clbi branch if signed <
blmi branch if result is negative
blne branch if result is nonzero
blpl branch if result is positive

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
reva Fuzzy logic rule evaluation
rewb 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
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
suba 8-bit sub from RegA
subb 8-bit sub from RegB
subd 16-bit sub from RegD
swi software interrupt, trap
tab transfer A to B
tap transfer A to CC
tba transfer B to A
tba transfer CC to A
wai wait for interrupt
wav weighted Fuzzy logic average
xgdx exchange RegD with RegX
xgdy exchange RegD with RegY

---

**STX**

Store Index Register X

Operation: \((X_H : X_L) \Rightarrow M : M + 1\)

<table>
<thead>
<tr>
<th>Source Form</th>
<th>Address Mode</th>
<th>Object Code</th>
<th>Access Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>STX opr8a</td>
<td>DIR</td>
<td>5E dd</td>
<td>PW</td>
</tr>
<tr>
<td>STX opr16a</td>
<td>EXT</td>
<td>7E hh 11</td>
<td>PWO</td>
</tr>
<tr>
<td>STX oprx0, xysp</td>
<td>IDX</td>
<td>6E xb</td>
<td>PW</td>
</tr>
<tr>
<td>STX oprx9, xysp</td>
<td>IDX1</td>
<td>6E xb ff</td>
<td>PWO</td>
</tr>
<tr>
<td>STX oprx16, xysp</td>
<td>IDX2</td>
<td>6E xb ee ff</td>
<td>PWP</td>
</tr>
<tr>
<td>STX [D, xysp]</td>
<td>[D,IDX]</td>
<td>6E xb ee ff</td>
<td>PIPW</td>
</tr>
</tbody>
</table>

**example**

<table>
<thead>
<tr>
<th>addressing mode</th>
<th>Effective Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>ldda #u</td>
<td>No EA</td>
</tr>
<tr>
<td>ldda u</td>
<td>EA is 8-bit address (0 to 255)</td>
</tr>
<tr>
<td>ldda U</td>
<td>EA is a 16-bit address</td>
</tr>
<tr>
<td>ldda m,r</td>
<td>EA=r+m (-16 to 15)</td>
</tr>
<tr>
<td>ldda v,+r</td>
<td>r=r+v, EA=r (1 to 8)</td>
</tr>
<tr>
<td>ldda v,-r</td>
<td>r=r-v, EA=r (1 to 8)</td>
</tr>
<tr>
<td>ldda v,r+</td>
<td>EA=r, r=r+v (1 to 8)</td>
</tr>
<tr>
<td>ldda v,r-</td>
<td>EA=r, r=r-v (1 to 8)</td>
</tr>
<tr>
<td>ldda A,r</td>
<td>EA=r+A, zero padded</td>
</tr>
<tr>
<td>ldda B,r</td>
<td>EA=r+B, zero padded</td>
</tr>
<tr>
<td>ldda D,r</td>
<td>EA=r+D</td>
</tr>
<tr>
<td>ldda q,r</td>
<td>EA=r+q (-256 to 255)</td>
</tr>
<tr>
<td>ldda W,r</td>
<td>EA=r+W (-32768 to 65535)</td>
</tr>
<tr>
<td>ldda [D,r]</td>
<td>EA=(r+D)</td>
</tr>
<tr>
<td>ldaa [W,r]</td>
<td>indirect</td>
</tr>
</tbody>
</table>

*Motorola 6812 addressing modes r is X, Y, SP, or PC*
<table>
<thead>
<tr>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
<th>Column 4</th>
<th>Column 5</th>
<th>Column 6</th>
<th>Column 7</th>
<th>Column 8</th>
<th>Column 9</th>
<th>Column 10</th>
<th>Column 11</th>
<th>Column 12</th>
<th>Column 13</th>
<th>Column 14</th>
<th>Column 15</th>
<th>Column 16</th>
<th>Column 17</th>
<th>Column 18</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0,X</td>
<td>10</td>
<td>16,X</td>
<td>5b const</td>
<td>6b const</td>
<td>20</td>
<td>1,X</td>
<td>5b const</td>
<td>2,Y</td>
<td>5b const</td>
<td>40</td>
<td>1,Y</td>
<td>5b const</td>
<td>60</td>
<td>1,Y</td>
<td>5b const</td>
<td>80</td>
</tr>
<tr>
<td>01</td>
<td>1,X</td>
<td>15,X</td>
<td>5b const</td>
<td>5b const</td>
<td>21</td>
<td>2,X</td>
<td>5b const</td>
<td>2,Y</td>
<td>5b const</td>
<td>41</td>
<td>1,Y</td>
<td>5b const</td>
<td>61</td>
<td>1,Y</td>
<td>5b const</td>
<td>81</td>
<td>1,Y</td>
</tr>
<tr>
<td>02</td>
<td>2,X</td>
<td>14,X</td>
<td>5b const</td>
<td>5b const</td>
<td>22</td>
<td>3,X</td>
<td>5b const</td>
<td>2,Y</td>
<td>5b const</td>
<td>42</td>
<td>3,Y</td>
<td>5b const</td>
<td>62</td>
<td>3,Y</td>
<td>5b const</td>
<td>82</td>
<td>3,Y</td>
</tr>
<tr>
<td>03</td>
<td>3,X</td>
<td>13,X</td>
<td>5b const</td>
<td>5b const</td>
<td>23</td>
<td>4,X</td>
<td>5b const</td>
<td>2,Y</td>
<td>5b const</td>
<td>43</td>
<td>4,Y</td>
<td>5b const</td>
<td>63</td>
<td>4,Y</td>
<td>5b const</td>
<td>83</td>
<td>4,Y</td>
</tr>
<tr>
<td>04</td>
<td>4,X</td>
<td>12,X</td>
<td>5b const</td>
<td>5b const</td>
<td>24</td>
<td>5,X</td>
<td>5b const</td>
<td>2,Y</td>
<td>5b const</td>
<td>44</td>
<td>5,Y</td>
<td>5b const</td>
<td>64</td>
<td>5,Y</td>
<td>5b const</td>
<td>84</td>
<td>5,Y</td>
</tr>
<tr>
<td>05</td>
<td>5,X</td>
<td>11,X</td>
<td>5b const</td>
<td>5b const</td>
<td>25</td>
<td>6,X</td>
<td>5b const</td>
<td>2,Y</td>
<td>5b const</td>
<td>45</td>
<td>6,Y</td>
<td>5b const</td>
<td>65</td>
<td>6,Y</td>
<td>5b const</td>
<td>85</td>
<td>6,Y</td>
</tr>
<tr>
<td>06</td>
<td>6,X</td>
<td>10,X</td>
<td>5b const</td>
<td>5b const</td>
<td>26</td>
<td>7,X</td>
<td>5b const</td>
<td>2,Y</td>
<td>5b const</td>
<td>46</td>
<td>7,Y</td>
<td>5b const</td>
<td>66</td>
<td>7,Y</td>
<td>5b const</td>
<td>86</td>
<td>7,Y</td>
</tr>
<tr>
<td>07</td>
<td>7,X</td>
<td>9,X</td>
<td>5b const</td>
<td>5b const</td>
<td>27</td>
<td>8,X</td>
<td>5b const</td>
<td>2,Y</td>
<td>5b const</td>
<td>47</td>
<td>8,Y</td>
<td>5b const</td>
<td>67</td>
<td>8,Y</td>
<td>5b const</td>
<td>87</td>
<td>8,Y</td>
</tr>
<tr>
<td>08</td>
<td>8,X</td>
<td>8,X</td>
<td>5b const</td>
<td>5b const</td>
<td>28</td>
<td>9,X</td>
<td>5b const</td>
<td>2,Y</td>
<td>5b const</td>
<td>48</td>
<td>9,Y</td>
<td>5b const</td>
<td>68</td>
<td>9,Y</td>
<td>5b const</td>
<td>88</td>
<td>9,Y</td>
</tr>
<tr>
<td>09</td>
<td>9,X</td>
<td>7,X</td>
<td>5b const</td>
<td>5b const</td>
<td>29</td>
<td>10,X</td>
<td>5b const</td>
<td>2,Y</td>
<td>5b const</td>
<td>49</td>
<td>10,Y</td>
<td>5b const</td>
<td>69</td>
<td>10,Y</td>
<td>5b const</td>
<td>89</td>
<td>10,Y</td>
</tr>
<tr>
<td>10</td>
<td>10,X</td>
<td>6,X</td>
<td>5b const</td>
<td>5b const</td>
<td>30</td>
<td>11,X</td>
<td>5b const</td>
<td>2,Y</td>
<td>5b const</td>
<td>50</td>
<td>11,Y</td>
<td>5b const</td>
<td>70</td>
<td>11,Y</td>
<td>5b const</td>
<td>90</td>
<td>11,Y</td>
</tr>
<tr>
<td>11</td>
<td>11,X</td>
<td>5,X</td>
<td>5b const</td>
<td>5b const</td>
<td>31</td>
<td>12,X</td>
<td>5b const</td>
<td>2,Y</td>
<td>5b const</td>
<td>51</td>
<td>12,Y</td>
<td>5b const</td>
<td>71</td>
<td>12,Y</td>
<td>5b const</td>
<td>91</td>
<td>12,Y</td>
</tr>
<tr>
<td>12</td>
<td>12,X</td>
<td>4,X</td>
<td>5b const</td>
<td>5b const</td>
<td>32</td>
<td>13,X</td>
<td>5b const</td>
<td>2,Y</td>
<td>5b const</td>
<td>52</td>
<td>13,Y</td>
<td>5b const</td>
<td>72</td>
<td>13,Y</td>
<td>5b const</td>
<td>92</td>
<td>13,Y</td>
</tr>
<tr>
<td>13</td>
<td>13,X</td>
<td>3,X</td>
<td>5b const</td>
<td>5b const</td>
<td>33</td>
<td>14,X</td>
<td>5b const</td>
<td>2,Y</td>
<td>5b const</td>
<td>53</td>
<td>14,Y</td>
<td>5b const</td>
<td>73</td>
<td>14,Y</td>
<td>5b const</td>
<td>93</td>
<td>14,Y</td>
</tr>
<tr>
<td>14</td>
<td>14,X</td>
<td>2,X</td>
<td>5b const</td>
<td>5b const</td>
<td>34</td>
<td>15,X</td>
<td>5b const</td>
<td>2,Y</td>
<td>5b const</td>
<td>54</td>
<td>15,Y</td>
<td>5b const</td>
<td>74</td>
<td>15,Y</td>
<td>5b const</td>
<td>94</td>
<td>15,Y</td>
</tr>
<tr>
<td>15</td>
<td>15,X</td>
<td>1,X</td>
<td>5b const</td>
<td>5b const</td>
<td>35</td>
<td>16,X</td>
<td>5b const</td>
<td>2,Y</td>
<td>5b const</td>
<td>55</td>
<td>16,Y</td>
<td>5b const</td>
<td>75</td>
<td>16,Y</td>
<td>5b const</td>
<td>95</td>
<td>16,Y</td>
</tr>
<tr>
<td>16</td>
<td>16,X</td>
<td>0,X</td>
<td>5b const</td>
<td>5b const</td>
<td>36</td>
<td>17,X</td>
<td>5b const</td>
<td>2,Y</td>
<td>5b const</td>
<td>56</td>
<td>17,Y</td>
<td>5b const</td>
<td>76</td>
<td>17,Y</td>
<td>5b const</td>
<td>96</td>
<td>17,Y</td>
</tr>
<tr>
<td>17</td>
<td>17,X</td>
<td>-1,X</td>
<td>5b const</td>
<td>5b const</td>
<td>37</td>
<td>18,X</td>
<td>5b const</td>
<td>2,Y</td>
<td>5b const</td>
<td>57</td>
<td>18,Y</td>
<td>5b const</td>
<td>77</td>
<td>18,Y</td>
<td>5b const</td>
<td>97</td>
<td>18,Y</td>
</tr>
<tr>
<td>18</td>
<td>18,X</td>
<td>-2,X</td>
<td>5b const</td>
<td>5b const</td>
<td>38</td>
<td>19,X</td>
<td>5b const</td>
<td>2,Y</td>
<td>5b const</td>
<td>58</td>
<td>19,Y</td>
<td>5b const</td>
<td>78</td>
<td>19,Y</td>
<td>5b const</td>
<td>98</td>
<td>19,Y</td>
</tr>
<tr>
<td>19</td>
<td>19,X</td>
<td>-3,X</td>
<td>5b const</td>
<td>5b const</td>
<td>39</td>
<td>20,X</td>
<td>5b const</td>
<td>2,Y</td>
<td>5b const</td>
<td>59</td>
<td>20,Y</td>
<td>5b const</td>
<td>79</td>
<td>20,Y</td>
<td>5b const</td>
<td>99</td>
<td>20,Y</td>
</tr>
<tr>
<td>20</td>
<td>20,X</td>
<td>-4,X</td>
<td>5b const</td>
<td>5b const</td>
<td>40</td>
<td>21,X</td>
<td>5b const</td>
<td>2,Y</td>
<td>5b const</td>
<td>60</td>
<td>21,Y</td>
<td>5b const</td>
<td>80</td>
<td>21,Y</td>
<td>5b const</td>
<td>100</td>
<td>21,Y</td>
</tr>
<tr>
<td>21</td>
<td>21,X</td>
<td>-5,X</td>
<td>5b const</td>
<td>5b const</td>
<td>41</td>
<td>22,X</td>
<td>5b const</td>
<td>2,Y</td>
<td>5b const</td>
<td>61</td>
<td>22,Y</td>
<td>5b const</td>
<td>81</td>
<td>22,Y</td>
<td>5b const</td>
<td>101</td>
<td>22,Y</td>
</tr>
<tr>
<td>22</td>
<td>22,X</td>
<td>-6,X</td>
<td>5b const</td>
<td>5b const</td>
<td>42</td>
<td>23,X</td>
<td>5b const</td>
<td>2,Y</td>
<td>5b const</td>
<td>62</td>
<td>23,Y</td>
<td>5b const</td>
<td>82</td>
<td>23,Y</td>
<td>5b const</td>
<td>102</td>
<td>23,Y</td>
</tr>
<tr>
<td>23</td>
<td>23,X</td>
<td>-7,X</td>
<td>5b const</td>
<td>5b const</td>
<td>43</td>
<td>24,X</td>
<td>5b const</td>
<td>2,Y</td>
<td>5b const</td>
<td>63</td>
<td>24,Y</td>
<td>5b const</td>
<td>83</td>
<td>24,Y</td>
<td>5b const</td>
<td>103</td>
<td>24,Y</td>
</tr>
<tr>
<td>24</td>
<td>24,X</td>
<td>-8,X</td>
<td>5b const</td>
<td>5b const</td>
<td>44</td>
<td>25,X</td>
<td>5b const</td>
<td>2,Y</td>
<td>5b const</td>
<td>64</td>
<td>25,Y</td>
<td>5b const</td>
<td>84</td>
<td>25,Y</td>
<td>5b const</td>
<td>104</td>
<td>25,Y</td>
</tr>
<tr>
<td>25</td>
<td>25,X</td>
<td>-9,X</td>
<td>5b const</td>
<td>5b const</td>
<td>45</td>
<td>26,X</td>
<td>5b const</td>
<td>2,Y</td>
<td>5b const</td>
<td>65</td>
<td>26,Y</td>
<td>5b const</td>
<td>85</td>
<td>26,Y</td>
<td>5b const</td>
<td>105</td>
<td>26,Y</td>
</tr>
</tbody>
</table>
(5) Question 1. What is the signed integer value (in decimal) of the 8-bit hexadecimal number $\text{C3}$?

Question 2. Consider the result of executing the following two 6812 assembly instructions.

```
ldab #100
subb #210
```

(5) Part a) What will be the value of the carry (C) bit?
(5) Part b) What will be the value of the overflow (V) bit?

(5) Question 3. A software variable can take on the following specific values -5.00, -4.99, -4.98, ..., 4.98, 4.99, 5.00. Which number format should be used for this variable? If more than one format could be used to solve the problem, choose the most space-efficient format. Enter the correct letter A-H.

A) 8-bit signed fixed-point number with resolution of 0.1
B) 8-bit signed fixed-point number with resolution of 0.01
C) 16-bit signed fixed-point number with resolution of 0.01
D) 16-bit unsigned fixed-point number with resolution of 0.001
E) 32-bit floating point
F) 8-bit signed integer
G) 16-bit signed integer
H) 32-bit signed integer

(5) Question 4. A certain ohmmeter has a range of 0 to $R_{\text{max}}$, a resolution of 0.1 $\Omega$, and a precision of 3½ decimal digits. What is $R_{\text{max}}$?

(5) Question 5. Show the machine code generated by the instruction

```
stx -5,y
```

(5) Question 6. The mul instruction multiplies the unsinged value in RegA by the unsigned value in RegB and stores the product in RegD. Give example input values (if any) that cause an overflow.

(5) Question 7. The idiv instruction divides the unsigned value in RegD by the unsigned value in RegX and stores the quotient in RegX. Give example input values (if any) that cause an overflow.

(5) Question 8. Consider the following program

```
$5000                                           org  $5000
$5000 CF4000          [ 2]{OP}main lds  #$4000
$5003 0707            [ 4]{PPPS}    bsr  Add1   ;part a)
$5005 CE1234          [ 2]{OP}    ldx  #$1234
$5008 5E02            [ 2]{PW}stx  2      ;part b)
$500A 183E            [16]{rOPw}Add1 inc  $0240
$500C 720240          [ 4]{rOPw}  stop
$500F 3D              [ 5]{UfPPP}rts
$FFFE                                           org  $FFFE
$FFFE 5000                                      fdb  main
```

(5) Part a) What value(s) is(are) pushed on the stack when the bsr Add1 instruction is executed?
(10) Part b) Show the simplified bus cycles occurring when the stx 2 instruction is executed. In the “changes” column, specify which registers get modified during that cycle, and the corresponding new values. Do not worry about changes to the CCR. Just show the one instruction.

(15) Question 9. Write assembly language instructions that make Port T bit 3 an input and Port T bit 2 an output. Full credit will be given to the code that modifies two bits in the direction register, without modifying the other six.

(30) Question 10. Write an assembly language subroutine, called Set5, which sets memory locations $3900$ to $397F$ (128 locations) equal to the value $05$. Full credit will be given to the code that implements a for-loop and uses the post-increment indexed addressing mode.