EE/CPE 421/521 – Microcomputer
Test I Solutions

Instructor: Dr. Aleksandar Milenkovic
Date: February 19, 2004
Place: EB 240
Time: 5:30 PM – 6:50 PM

Note: Work should be performed systematically and neatly. This exam is closed books and closed neighbour(s). Allowable items include exam, pencils, straight edge, calculator, and materials distributed by the instructor. Best wishes.

<table>
<thead>
<tr>
<th>Question</th>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>Sum</td>
<td>105</td>
<td></td>
</tr>
</tbody>
</table>

Please print in capitals:
Last name:____________________________
First name: ___________________________
1. (10 points, Misc)
A. (6 points) Draw a word-wide HEXADECIMAL content of memory cells corresponding to the following sequence of assembler directives:

```
ORG $2500
A  DS.B 2
P  EQU 20
V1 DC.W 10,15
V2 DC.L $40302010
V3 DC.W P+5
V4 DC.L -3
```

<table>
<thead>
<tr>
<th>Address [Hex]</th>
<th>Content &lt;15:0&gt; [Hex]</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>2500</td>
<td>----</td>
<td></td>
</tr>
<tr>
<td>2502</td>
<td>000A</td>
<td></td>
</tr>
<tr>
<td>2504</td>
<td>000F</td>
<td></td>
</tr>
<tr>
<td>2506</td>
<td>4030</td>
<td></td>
</tr>
<tr>
<td>2508</td>
<td>2010</td>
<td></td>
</tr>
<tr>
<td>250a</td>
<td>0019</td>
<td></td>
</tr>
<tr>
<td>250c</td>
<td>FFFF</td>
<td></td>
</tr>
<tr>
<td>2510</td>
<td>FFFB</td>
<td></td>
</tr>
</tbody>
</table>

B. (4 points) Motorola 68000 microprocessor has:

(i) __8__ address registers  
(ii) __8__ data registers  
(iii) register size is __32__ bits  
(iv) status bit V represents ____Overflow______. It is set when __there is an overflow (result of operation is not correct).__
2. (20 points) Represent the state of the stack and values of SP and A6 during the execution of the following program:

```c
void main(void) {
    int coef, x, y;
    coef = 5; x = 3;
    mult(coef,&x);  /* evaluate function y=coef*x */
}
void mult(int a, int *b) {
    int c = 5;
    *b = c*a;
}
```

Code generated by the cross compiler is given below:

```c
*1  void mult(int a, int *b) {
* Parameter a is at 8(A6)
* Parameter b is at 10(A6)
* Variable c is at -2(A6)
    _mult
        LINKA6,#-2
*2  int c = 5;
    MOVE#5,-2(A6)
*3  *b = c*a;
    MOVE8(A6),D1
    MOVE.D1,D0
    LSL #2,D1
    ADD D0,D1
    MOVEA.L 10(A6),A4
    MOVE.D1,(A4)
*4  }
    UNLKA6
    RTS
*5  void main(void) {
* Variable x is at -2(A6)
* Variable y is at -4(A6)
    _main
        LINKA6,#-4
*6  int x, y;
    x = 3;
    MOVE#3,-2(A6)
*7  mult(x, &y);
    PEA.L -4(A6)
    MOVE#3,-(A7)
    JSR _mult
*8  }
    UNLKA6
    RTS
    END
```

The state of the stack and values of SP and A6 during the execution of the program are as follows:

- **Initial State:**
  - SP: 3
  - A6: 0
  - Old A6: 0
  - Old TOS: 0
  - y = 0
  - x = 0

- **After** `coef = 5; x = 3;`:
  - Old A6: 0
  - Old TOS: 0
  - y = 0
  - x = 0
  - SP: 3
  - A6: 0

- **After** `mult(coef,&x);`:
  - Old A6: 0
  - Old TOS: 0
  - y = 15
  - x = 3
  - SP: 3
  - A6: 0

- **After** `main(void)`:
  - Old A6: 0
  - Old TOS: 0
  - y = 15
  - x = 3
  - SP: 3
  - A6: 0
3. (20 points) For the given fraction of assembly language program:

L1
MOVE.B 8 (A6), D0  12
CLR.L D1  6
MOVEQ #3, D5  4

L2
LSL.W #1, D0  6+2*1=8
ROXL.W #1, D1  6+2*1=8
DBF D5, L2  10/14 if expired
RTS  16

Loop, executed 4 times

A. (10 points) Find the total execution time of the given program on an 8 MHz 68000 microprocessor.

At 8MHz, cycle time is $T_{cycle} = \frac{1}{8MHz} = 125$ nsec

Clock cycles: $C = 12 + 6 + 4 + 3 \cdot (8 + 8 + 10) + 1 \cdot (8 + 8 + 14) + 16 = 146$

$T_{exe} = T_{cycle} \times C = 125 \text{nsec} \times 146 = 18.25 \mu s$

B. (5 points) Calculate the average CPI (number of clocks per instructions).

Total number of instructions executed:

$N = 3 + 4 \cdot 3 + 1 = 16$

$CPI = \frac{C}{N} = \frac{146}{16} = 9.125$

C. (5 points) Calculate the MIPS rate.

MIPS rate $= 10^{-6} \times f / CPI = 8 / 9.125 = 0.877 \text{ MIPS}$
4. (20 points)
A. (5 points) Why it’s a good idea to pass parameters to and from a subroutine by means of the stack? Compare pros and cons for passing parameters via the stack, registers, and explicit memory locations.

Using explicit memory locations for passing parameters could be a bad idea. Let’s say we pass a parameter through a memory location and call a corresponding subroutine. While we are in the subroutine an interrupt occurs and it is accepted. What happens if the interrupt handling routine calls the same subroutine \(\Rightarrow\) our initial parameter will be overwritten.

Using registers to pass parameters is safer, assuming that you save working registers on the stack. However, there is a limited number of registers.

Passing parameters on the stack is the best because a subroutine can be interrupted and called by another subroutine without corrupting the parameters.

B. (15 points) Write a subroutine using 68K assembly that swaps elements of a byte array, that is the first element of the array \(a(0)\) is swapped with the last element \(a(n-1)\), \(a(1)\) with \(a(n-2)\), etc. Assume that parameters for subroutine swap_array(short int *a, int n), the starting address and array size, are prepared on the stack in the main program. Use registers for local variables in the subroutine.

```
SwapArray
    LINK A6,0
    MOVEM.L A0-A1/D0-D1,-(A7)
    MOVE.L +10(A6),A0       * A0 keeps the starting address
    MOVE.L +8(A6), D0       * D0 keeps the size
    ADDA D0,A1              * A1 points at the end+1
    ASR #1,D0               * divide size by 2
Loop:  MOVE.B (A0),D1
    MOVE.B -(A1),(A0)+
    MOVE.B D1, (A1)
    SUBQ.W #1,D0
    BNE Loop
    MOVEM.L -(A7),A0-A1/D0-D1
    RTS
```
5. **(30 points)** What is the effect of applying each of the following 68000 instructions assuming the initial conditions shown below? Represent modified internal registers, memory locations and condition codes.

(a) **MOVE.B** (A1)+, D0  
**CMPL.B** #$5A, D0

\[
\begin{align*}
A1 &= $00007028 \\
D0<7:0> &= [M($7028)] = $79 \\
\text{After MOV.B: } D0 &= $01234579 \\
A1 &= $7029 \\
\end{align*}
\]

(b) **CMPM.B** (A1)+, (A3)+

\[
\begin{align*}
(A1) &= [M($7028)] = $79 \\
(A3) &= [M($7030)] = $2B \\
\{\text{New contents: } A1 = $7029, A3 = $7031\} \\
\end{align*}
\]

(c) **MOVE.L** 5(A3,D6.W), D4

\[
\begin{align*}
A3 &= $7030 \\
D6.W &= $0003 \\
\text{+ } $0005 \\
EA &= $7038 \\
\{D4 = $214255EA\} \\
\end{align*}
\]

(d) **LINK** A5,#-4

\[
\begin{align*}
(A7) &= $10080 \\
\text{Store A5 on the top, and A5 is new frame pointer, allocate 4 bytes}\ \\
(A7) &= $10080 - $8 = $10078 \ (4 \text{ bytes for old A5, 4 bytes for frame}) \\
\text{mem}[1007C] &= $00 \\
\text{mem}[1007D] &= $00 \\
\text{mem}[1007E] &= $FF \\
\text{mem}[1007F] &= $FA \\
\end{align*}
\]
(e) AND.B 6(A5), D4

\[
\begin{align*}
A5 &= \$FFFA \\
D4 &= \$33449127, \ D4.B &= \$27 \\
+ &\ \frac{0006}{10000} \ %0101 1111 \\
\end{align*}
\]

\[
M[10000]=\$DD \quad \%0000 0101 = \$05 \Rightarrow D4 = \$33449105
\]

<table>
<thead>
<tr>
<th>X</th>
<th>N</th>
<th>Z</th>
<th>V</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(f) BTST.B #3, (A3)

\[
A3 = \$7030 \\
[M(\$7030)] = \$2B = \%10110111 \\
\text{Tested bit was NOT zero } \Rightarrow Z=0
\]

<table>
<thead>
<tr>
<th>X</th>
<th>N</th>
<th>Z</th>
<th>V</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(g) ASL.B #1, D5

\[
D5 = \text{AAAAAA} \frac{10101010}{%1010 1010} \quad 0 = \%01010100 = \$54 \\
\text{C}
\]

\[
\begin{align*}
\text{D5} &= \text{AAAAAA54} \\
\text{V} &= 1
\end{align*}
\]

(h) MOVE.B $A7(A7), (A3)+

\[
\begin{align*}
EA &= \$00010080 \\
+ &\ \frac{FFFFFFFA7}{00010027} \quad \text{(sign extended } A7 = FFFFFFA7) \\
\end{align*}
\]

\[
\begin{align*}
[M(\$10027)] &= \$2D \quad [M(\$7030)] \leftarrow \$2D \\
A3 &= \$7030 \\
A3 &= \$7031
\end{align*}
\]

<table>
<thead>
<tr>
<th>X</th>
<th>N</th>
<th>Z</th>
<th>V</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
### 68000 Registers

<table>
<thead>
<tr>
<th>D0</th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>D4</th>
<th>D5</th>
<th>D6</th>
<th>D7</th>
</tr>
</thead>
<tbody>
<tr>
<td>01234567</td>
<td>89ABCDEF</td>
<td>0001002D</td>
<td>ABCD7FFF</td>
<td>33449127</td>
<td>AAAAAAAA</td>
<td>ABCD0003</td>
<td>55555555</td>
</tr>
<tr>
<td>0007020</td>
<td>0007028</td>
<td>0001000A</td>
<td>0007030</td>
<td>00010020</td>
<td>000FFFA</td>
<td>00010000</td>
<td>00010080</td>
</tr>
</tbody>
</table>

### Status register

2700

### Main memory

| 007000 | 007001 | 007002 | 007003 | 007004 | 007005 | 007006 | 007007 | 007008 | 007009 | 00700A | 00700B | 00700C | 00700D | 00700E | 00700F | 007010 | 007011 | 007012 | 007013 | 007014 | 007015 | 007016 | 007017 | 007018 | 007019 | 00701A | 00701B | 00701C | 00701D | 00701E | 00701F |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| AE    | F2    | 32    | 77    | 89    | 90    | 1A    | AE    | EE    | F1    | F2    | A4    | AE    | 88    | 88    | AA    | E4    | 7E    | 8D    | 9C    | C4    | B2    | 12    | 39    | 90    | 00    | 89    | 14    | 01    | 3D    | 77    | 89    | 9A    | 070020 | 070021 | 070022 | 070023 | 070024 | 070025 | 070026 | 070027 | 070028 | 070029 | 07002A | 07002B | 07002C | 07002D | 07002E | 07002F | 070030 | 070031 | 070032 | 070033 | 070034 | 070035 | 070036 | 070037 | 070038 | 070039 | 07003A | 07003B | 07003C | 07003D | 07003E | 07003F |