Department of Electrical and Computer Engineering
University of Alabama in Huntsville

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.

Question	Points	Score
1	10	
2	25	
3	20	
4	20	
5	30	
Sum	105	

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
```

Address [Hex]	Content <15:0> [Hex]	Comment
2500	**----**	
2502	**000A**	
2504	**000F**	
2506	**4030**	
2508	**2010**	
250A	**0019**	
250C	**FFFF**	
2510	**FFFB**	

B. (4 points) Motorola 68000 microprocessor has:

(i)___**8**__address registers

(ii)___**8**__ data registers

(iii) register size is ___32__ bits

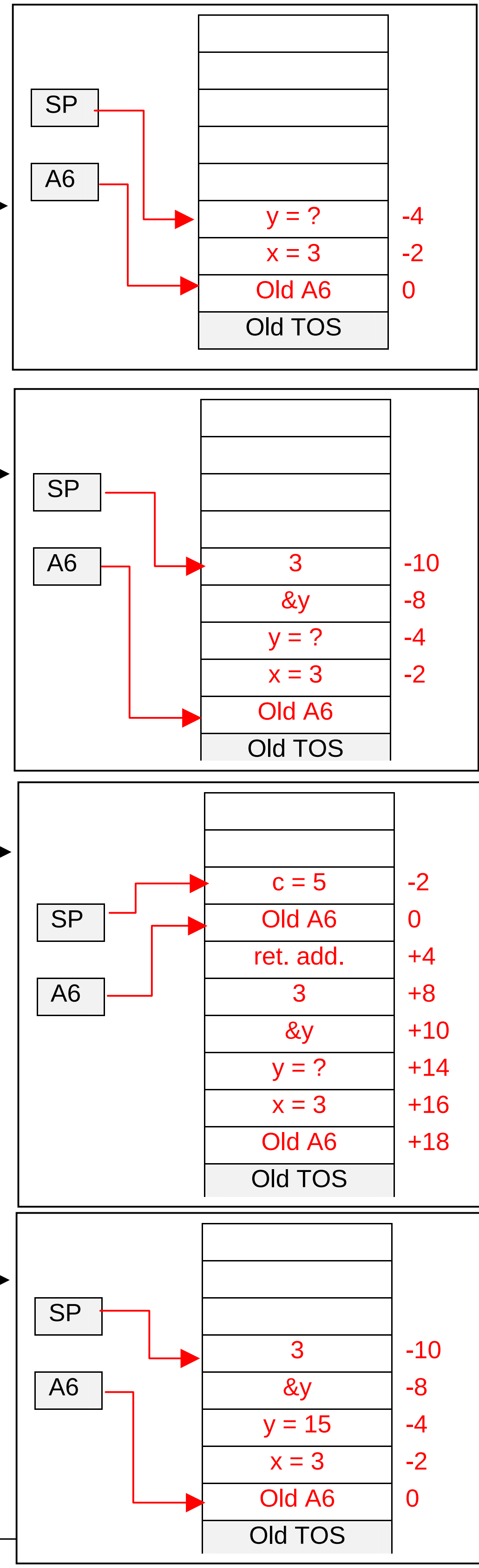
(iv) status bit V represents __**Overflow**______**.** It is set when **there is an overflow (result of operation is not correct).**

2. (20 points) Represent the state of the stack and values of SP and A6 during the execution of the following program:

```
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:

```
*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
        MOVED1,D0
        LSL  #2,D1
        ADD  D0,D1
        MOVEA.L   10(A6),A4
        MOVED1,(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;
*7          x = 3;
        MOVE#3,-2(A6)

*8          mult(x, &y);
        PEA.L     -4(A6)
        MOVE#3,-(A7)

        JSR  _mult
*9      }
        UNLKA6
        RTS
        END
```



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              } Loop,
      ROXL.W   #1,D1        6+2*1=8              } executed
      DBF      D5,L2        10/14 if expired     } 4 times
      RTS                   16
```

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}=1/8MHz=125nsec

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

$T_{exe} = T_{cycle} \times C$ = 125nsec × 146 = 18.25μs

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

Total number of instructions executed:

N=3+4*3+1=16

CPI = C/N = 146/16 = 9.125

C. (5 points) Calculate the MIPS rate.

MIPS rate = 10^{-6} * f / CPI = 8 / 9.125 = 0.877 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 => 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
CMPI.B #$5A, D0

A1 = $00007028
D0<7:0> = [M($7028)] = $79
after MOVE.B: D0 = $01234579
A1 = $7029

After CMPB.I

Initial XNZVC = 00000
$79 - $5A = $1F ⇒ XNZVC = 00000

X	N	Z	V	C
0	0	0	0	0

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

(A1) = [M($7028)] = $79
(A3) = [M($7030)] = $2B
$2B - $79 ⇒ $B2
New contents: A1=$7029
A3=$7031

X	N	Z	V	C
0	1	0	0	1

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

A3 = $7030
D6.W = $0003
+ $0005
EA = $7038

[M($7038)] = $21
[M($7039)] = $42
[M($703A)] = $55
[M($703B)] = $EA

D4 = $214255EA

X	N	Z	V	C
0	0	0	0	0

(d) LINK A5,#-4

(A7) = $10080

Store A5 on the top, and A5 is new frame pointer, allocate 4 bytes

(A7) = $10080 – $8 = $10078 (4 bytes for old A5, 4 bytes for frame)

mem[$1007C] = $00

mem[$1007D] = $00

mem[$1007E] = $FF

mem[$1007F] = $FA

X	N	Z	V	C
0	0	0	0	0

(e) AND.B 6(A5), D4

```
A5 =      $FFFA          D4 = $33449127, D4.B = $27
       +  $0006                     %0010 0111
          $10000               AND  %1101 1101
M[10000]=$DD                        %0000 0101 = $05 => D4 = $33449105
```

X	N	Z	V	C
0	0	0	0	0

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

```
                                   bit #3
A3 = $7030
[M($7030)] = $2B = %00101011     Tested bit was NOT zero ⇒ Z=0
```

X	N	Z	V	C
0	0	0	0	0

(g) ASL.B #1, D5

```
D5 = AAAAAAAA          ASL: X    10101010 ← 0 = %01010100 = $54
$AA = %1010 1010            C

        $AA - negative }
        $54 - positive }  V = 1
D5 = $AAAAAA54
```

X	N	Z	V	C
1	0	0	1	1

(h)

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

```
EA =      $00010080
       +  $FFFFFFA7    (sign extended $A7 = $FFFFFFA7)
          $00010027

[M($10027)] = $2D }        [M($7030)] ← $2D
A3 = $7030        }    A3 = $7031
```

X	N	Z	V	C
0	0	0	0	0

68000 Registers

D0	01234567	D1	89ABCDEF	D2	0001002D	D3	ABCD7FFF
D4	33449127	D5	AAAAAAAA	D6	ABCD0003	D7	55555555
A0	00007020	A1	00007028	A2	0001000A	A3	00007030
A4	00010020	A5	0000FFFA	A6	00010000	A7	00010080
	Status register		2700				

Main memory

007000	AE	007020	5A	010000	DD	010020	DC
007001	F2	007021	AD	010001	B2	010021	25
007002	32	007022	99	010002	00	010022	15
007003	77	007023	92	010003	15	010023	17
007004	89	007024	79	010004	76	010024	29
007005	90	007025	33	010005	19	010025	39
007006	1A	007026	97	010006	92	010026	49
007007	AE	007027	14	010007	26	010027	2D
007008	EE	007028	79	010008	17	010028	B2
007009	F1	007029	E7	010009	14	010029	62
00700A	F2	00702A	00	01000A	23	01002A	81
00700B	A4	00702B	0A	01000B	E7	01002B	21
00700C	AE	00702C	88	01000C	19	01002C	45
00700D	88	00702D	18	01000D	92	01002D	18
00700E	AA	00702E	82	01000E	19	01002E	31
00700F	E4	00702F	79	01000F	54	01002F	D9
007010	7E	007030	2B	010010	45	010030	AA
007011	8D	007031	17	010011	99	010031	77
007012	9C	007032	46	010012	15	010032	78
007013	C4	007033	9E	010013	43	010033	AE
007014	B2	007034	FC	010014	25	010034	EA
007015	12	007035	FF	010015	76	010035	34
007016	39	007036	77	010016	89	010036	25
007017	90	007037	60	010017	17	010037	17
007018	00	007038	21	010018	81	010038	15
007019	89	007039	42	010019	17	010039	14
00701A	14	00703A	55	01001A	4E	01003A	17
00701B	01	00703B	EA	01001B	72	01003B	F9
00701C	3D	00703C	61	01001C	33	01003C	8A
00701D	77	00703D	81	01001D	23	01003D	0F
00701E	89	00703E	C9	01001E	E1	01003E	F2
00701F	9A	00703F	AA	01001F	CD	01003F	E5