

Midterm Oct 2003

1. [20 marks]

A. General 68K Assembler

Given the following data, what is the result of the operations on it?

Registers:

d0 = \$98237654	d1 = \$A68B3237	d2 = \$000300FF	d5 = \$0D358B45
d6 = \$00000010	d7 = \$FF00F2D0	a0 = \$00000001	a7 = \$00001000

Memory:

Addr	Data
\$00000000	\$F0394124
\$00000004	\$FF000087
\$00000008	\$00000004
\$0000000C	\$00000002
\$00000010	\$B4FE8224

Indicate all memory locations and registers that change values after the following instructions are executed, and the new values.

Eg.

and.l d2,d0

Result: d0 = \$00030054

If the instruction is illegal or the result cannot be determined with the information given, clearly state the reason. All questions are independent of each other and use the initial values above.

Indicate values in memory as M[\$addr] = \$xxx and show longwords as in the tables above. For (e) show results for both instructions.

- a) or.b d7,3(a0) M[\$00000004]=\$FF000087 (or 'no changes')
- b) add.l \$8,\$C illegal (multiple Absolute modes not supported)
- c) add.b d0, d5 d5=\$0D358B99
- d) and.w a0,d1 illegal, a0 cannot be used in and instruction
- e) moveq.l #3,d3 d3=\$00000003
 pea.l 0(a0,d3) a7=\$00000FFC, M[\$00000FFC]=\$00000004

B. Branches

Is this branch taken or not? What condition codes are involved in the branch decision and what is its value/are their values? ['Taken' means that the branch offset is used and we end up at MyLabel]

move.l #00008901,d0
add.w #ABCD7A43,d0
bclr.l #2,d0
beq MyLabel

Taken Not Taken

Condition Code(s) & Value(s): Z=0

C Instruction Sizes

Fill in the blank areas in the following table. The memory accesses column should include all memory accesses to get the instruction, the operands and to store the result. Assume each memory access can transfer at most one word.

Instruction	# of words in entire instruction	# of memory accesses
add.w d0,1550	3 (or 2)	5 (or 4)
add.w #1550,d0	2	2
addq.w #3,d2	1	1
move.b (a2),(a3)+	1	3

Bracketted values for
short addressing

2. [20 marks] Consider the assembly in (i), which implements the C-code factorial function in (ii).

```

                                org      $3000
(i) 003000: FACT   link      a6,#0
     003004:      tst.l     8(a6)
     003008:      bne      NEXT
     00300c:      move.l   #1,d0
     003012:      bra      DONE

     003016: NEXT   move.l   8(a6),d0
     00301a:      subq.l   #1,d0
     00301c:      move.l   d0,-(a7)
     00301e:      bsr      FACT
     003022:      addq.l   #4,a7
     003024:      mulu.w   8(a6),d0

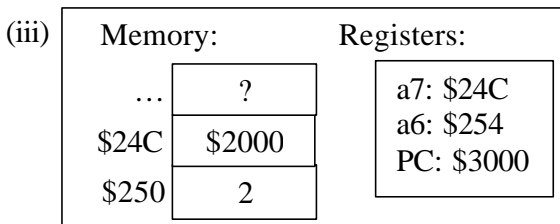
     003028: DONE   unlk     a6
     00302a:      rts

```

```

(ii) int factorial(int x){
      if (x == 0){
          return 1;
      }
      return x * factorial(x-1);
}

```



Consider a call to factorial() where x is 2. The appropriate state of memory and registers at the point just before the link instruction executes for the first time is shown in (iii).

Your task: fill in the memory table and the two registers below, showing the configuration just before the unlk instruction executes for the *first* time. Also write one of the following descriptions to the right of each memory table entry: “return address”, “local variable”, “parameter x”, “callee-saved register”, “caller-saved register”, or “old a6”.

HINTS: 1) d0 is used for the return value; 2) FACT calls itself recursively, but this behaves no differently than one subroutine calling another; 3) Use the extra page following to trace the execution and the stack starting with the link instruction, then fill in the memory table on this page when you are happy with your answer.

Address	Value	Description	Registers
\$214			
\$218			
\$21C			a7: <u> \$230 </u>
\$220			
\$224			
\$228			a6: <u> \$230 </u>
\$22C			
\$230	\$23C	old a6	
\$234	\$3022	return address	
\$238	0	parameter x	
\$23C	\$248	old a6	
\$240	\$3022	return address	
\$244	1	parameter x	
\$248	\$254	old a6	
\$24C	\$2000	return address	
\$250	2	parameter x	

3. [25 marks]

Student information is being held in a data area, where each student record has the following format:

The first nine bytes are the student number, held in ASCII

The next byte is the course mark

The next word is the section identifier

There are well over three hundred such student records that have been loaded sequentially into memory starting at address \$10000. The last record loaded is a dummy record with a section identifier of \$FFFF, to show the end of the records.

- a. If a2 has the address of a student record, if using indexed addressing mode, what is the index value X such that X(a2) addresses the course mark?

$$X = \underline{9}$$

- b. If a2 has the address of a student record, what is the index value Y such that Y(a2) addresses the section identifier?

$$Y = \underline{10 \text{ or } \$0A}$$

- c. If a2 has the address of a student record, what is the location of the next record in the array / data area:

$$[a2] + \underline{12 \text{ or } \$0C}$$

- d. Write an assembler subroutine **GetMax** that will scan the entire list and find the highest mark. If there is a tie, the first student in the list with the highest mark should be found. The subroutine should return the starting address of this record in address register a3. Hints: (i) think about 3a – 3c above. (ii) Pseudocode will help the markers understand what you are trying to do and could lead to more part marks.

```
GetMax    movea.l #$10000,a0      ;point a0 to records
          movea.l a0,a3        ;init return pointer
          clr.b d0             ;d0 holds the max mark
Loop      move.w $A(a0),d1     ;get section ID
          cmpi.w $FFFF,d1     ;see if done
          beq Exit            ;leave loop if so
          cmp.b 9(a0),d0      ;see if max so far
          bge L1              ;br if not
          movea.l a0,a3        ;reset the pointer
          move.b 9(a0),d0      ;set the new max so far
L1        adda.l #$C,a0        ;go to next record
          bra Loop            ;and look at that one
Exit      rts                 ;done
```