UNIVERSITY OF DUBLIN

TRINITY COLLEGE

Faculty of Engineering and Systems Sciences

Department of Computer Science

B.A.(Mod.) Computer Science
Junior Freshman Examination

Trinity Term 2000

1BA3 - Introduction to Computing

Friday 2nd June | Luce Hall | 14.00-17.00

Michael Manzke

Attempt **FIVE** questions
(to be accompanied by a 68332 instruction set booklet and
`adda` instruction template sheet)

1. a) Discuss Memory Mapping.

   b) Design a chip select circuit for a CPU with 24 address pins that implement a device with the following specification:
   - 8 bit device
   - 8 register select pins
   - 256 registers

   Show your design as a digital logic symbol diagram. The device address space should start at `$32000`.

   c) Write an assembly language program that writes `#$10` into the twenty-second register of the device implemented in 1.b).

   d) Design and implement an assembly language program that increments a counter at memory location `$16000` every 2 seconds. You should use the PIT interrupt handling facility for this task (assuming EXTAL clock frequency = 32.768kHz).

2. a) What is the condition code register (CCR)? Demonstrate with examples, the use of the Z and N bits of the CCR.

b) Design and implement an assembly language program that calculates the even parity over the 7 MSBs (bit-7 to bit-1) of a byte at memory location `$2000` and insert the parity bit into the LSB (bit-0). You should use flow control for this task.

c) Design and implement an assembly language program that extends the parity program implemented in 2.b). The program should write:

- the string `'EVEN'` into memory locations starting at `$2004` if an even number of bits was set in location `$2000` prior to the insertion of the parity bit
- and the string `'ODD'` if the number of bits set was not even.

3. a) What is a Stack?

b) Describe the implementation of stacks on the MC68332.

c) Design and implement an assembly language program that has two subroutines. The first subroutine is called with the `jsr` instruction from the main program. The first subroutine then calls the second subroutine (nested) and subsequently returns to the first subroutine with the `rts` instruction. After that the first subroutine also uses the `rts` instruction to return to the main program. Explain the stack operations for each of the above steps.

4. a) Distinguish between row order and column order in the context of 2-dimensional arrays.

b) Discuss the calculation of appropriate index values for n-dimensional arrays. Provide formulas for the following three arrays: 2D array[y,z], 3D array[x,y,z] and 4D array[w,x,y,z].

c) Write an assembly language program that implements the following HLL subroutine using stack frames and frame pointers. To illustrate your solution, draw a diagram of the system stack contents when the statement >> temp = input1; << is executed. Assume that input1 and input2 will have been pushed onto the stack by the calling program.

```
EXCHANGE(WORD input1, WORD input2)
{
      WORD temp;

      temp = input1;
      input1 = input2;
      input2 = temp;
}
```

5. a) Given the MC68332 `adda` instruction template determine the full machine code (in hexadecimal) for the following instructions (include all required operands) and explain the four addressing modes.

   i) `adda.l #$1200,a5`
   ii) `adda.w a5,a2`
   iii) `adda.l (a2)+,a6`
   iv) `adda.w 8(a6),a4`

   b) Write a short assembly language program that will change the Opmode field of the `adda` instruction located at `$4000` in memory to word size, regardless of its previous contents.

   c) Explain the sign-extended word operation is this context.

6. a) What is an interrupt handler?

   b) Give a detailed description of the MC68332 interrupt handling mechanism.

   c) Implement a trap handler that will return the address of the trap instruction that caused the handler to be executed.

7. a) Discuss two methods of encoding negative numbers in the binary system. Explain the advantages and disadvantages of each.

   b) Distinguish between carry and overflow and give examples of arithmetic operations that will set each of the flags

   c) Design and implement an assembly language program that changes the sign of a signed binary number for each of the methods discussed in part 7a).

Operation: Source + Destination → Destination

Assembler Syntax: ADDA ⟨ea⟩ An

Attributes: Size = (Word, Long)

Description: Adds the source operand to the destination address register and stores the result in the address register. The entire destination address register is used regardless of the operation size.

Condition Codes:

Not affected

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	REGISTER			OPMODE			EFFECTIVE ADDRESS					
										MODE			REGISTER		

Instruction Fields:

Register field — Specifies any of the eight address registers. This is always the destination.

Opmode field — Specifies the size of the operation:

011 — Word operation. The source operand is sign-extended to a long operand and the operation is performed on the address register using all 32 bits.

111 — Long operation.

Effective Address field — Specifies source operand. All addressing modes are allowed as shown:

Addressing Mode	Mode	Register		Addressing Mode	Mode	Register
Dn	000	Reg. number: Dn		(xxx).W	111	000
An	001	Reg. number: An		(xxx).L	111	001
(An)	010	Reg. number: An		#⟨data⟩	111	100
(An) +	011	Reg. number: An				
– (An)	100	Reg. number: An				
(d_{16}, An)	101	Reg. number: An		(d_{16}, PC)	111	010
(d_8, An, Xn)	110	Reg. number: An		(d_8, PC, Xn)	111	011
(bd, An, Xn)	110	Reg. number: An		(bd, PC, Xn)	111	011