Imperial College of Science, Technology and Medicine	University of London
Computer Science (CS) / Software Engineering (SE)	BEng and MEng Examinations Part I
Department of Computing	Integrated Laboratory Course

Laboratory work is a continuously assessed part of the examinations
and is a required part of the degree assessment.
Laboratory work must be handed in for marking by the due date.
Late submissions may not be marked.

Exercise: 12	Working: Individual
Title: 80386 Assembly	
Issue date: 2nd February 2004	Due date: 9th February 2004
System: Windows 2000	Language: 80386 Assembler

Aims

- To gain experience in writing and testing 80386 assembler code using the assembler **a386** and the debugger **d386** available on the PCs running Windows 2000.

Background reading

- Further details on the assembler **a386** and the debugger **d386** can be found on the web at **http://www.doc.ic.ac.uk/~ ih/teaching/lectures/comparch/**.
- General information on the 80386 and related architectures is available at **http://developer.intel.com/design/pentiumii/manuals/**.

The Problem

- Write an 80386 method **matrix_print** which prints out in tabular form a matrix object residing at a specified location in memory. The matrix is stored in the format: number of rows, number of columns, and then the elements, row by row. The number-of-rows field, number-of-columns field and the elements are all 32-bit integers.
- Write an 80386 method **matrix_mult** which computes the product of two matrices at specified locations in memory, storing the product matrix at a third specified location in memory. The matrix storage format is as explained in **matrix_print** above.
- Your methods should be written in the file **matmult.asm**, which is provided as a *skeleton* initially (see below).

- As 80386 assembler is a low-level language, **your code should be well-commented**. The comments should explain the algorithm you use, either in English or in pseudo-code, and should clearly show which part does what.
- The Java pseudo-code for the two methods that you will need to translate are given below.

```
;
; class matrix {
;     int ROWS              // ints are 32-bit
;     int COLS
;     int elem [ROWS][COLS]
;
;     void print () {
;         output.newline ()
;         for (int row=0; row < this.ROWS; row++) {
;             for (int col=0; col < this.COLS; cols++) {
;                 output.tab ()
;                 output.int (this.elem[row, col])
;             }
;             output.newline ()
;         }
;     }
;
;     void mult (matrix A, matrix B) {
;         for (int row=0; row < this.ROWS; row++) {
;             for (int col=0; col < this.COLS; cols++) {
;                 int sum = 0
;                 for (int k=0; k < A.COLS; k++)
;                     sum = sum + A.elem[row, k] * B.elem[k, col]
;                 this.elem [row, col] = sum
;             }
;         }
;     }
; }
```

Submit by Monday 9th February 2004

What To Do

- As the **a386** and **d386** programs run under Windows you should log onto Windows on one of the dual boot machines in rooms 219, 202 or 203. Please make sure that your password is the same for both the Linux and Windows machines. This will allow you to mount your Linux home directory as **Drive H** under Windows next time you login. To start up the **a386** assembler you should select the start icon with the mouse which should pop up a menu. Keep the mouse pressed and select **Programs** and then **Lab**

and then **A386-D386**. This should open an **MSDOS** window in drive H. You will see the following message in the window:

```
                    .
                    .
                    .
To use A386, you must first make a source file with your text editor.
Invocation syntax: A386 srclist [TO] [obj] [sym] [lst] [xrf]
  e.g. A386 MYPROG.ASM MYPROG.COM
  or   A386 MYPROG.ASM MYPROG.OBJ

H:\>
```

You will now be in an environment where you can write, assemble, run and debug 80386 programs.

- Copy the skeleton file **matmult.asm** into your current working directory, this will be your home directory unless you have created a special directory for your assembly language programs. This file is stored at:

  `http://www.doc.ic.ac.uk/lab/firstyear/lab_area/GiveToStudents/matmult.asm`

  You can copy this file using your favourite web browser. If you are using Netscape on the PC, open the above location. Then select **File** followed by **Save As**. A **Save As...** window will then appear. **Select** drive H and then click on **Save** and the file will be copied into your current directory.

  Alternatively you can log onto Linux and copy the skeleton file using the command **exercise 12**.

- **matmult.asm** is a program which calls the methods **matrix_print** and **matrix_mult**, passing as parameters the addresses of the relevant matrix or matrices. The two input matrices are printed first, then their product is computed and finally the product is printed. Examine the file using the command **type**. At the moment these two methods simply return. Your task is to fill them in with code for performing matrix printing and matrix multiplication respectively.

- Write your assembly language program by editing the skeleton file **matmult.asm** using a text editor. A simple text editor with a similar look and feel to the *nedit* editor is *edit*. You can invoke this with the command **edit**. (Use **ALT f** to select the menu and the arrow keys to choose a menu option.)

- Once you have written your program you can assemble (compile) it using the *a386* command:

  ```
  a386 matmult.asm
  ```

  If there are errors they will be indicated in the file below the line with the error. You will need to re-edit and re-assemble the file. When the program is successfully assembled, the executable object code file **matmult.com** will be created.

- Execute the object code file by typing **matmult** at the DOS prompt. The two starting matrices and your computed product matrix should then appear in tabular form on the standard output.

- If there are problems with your code you can examine it while it is running using the debugger **d386**. This allows you to single step through the program, to inspect the registers and memory and many other things.

  Start the 80386 debugger using the *d386* command:

  ```
  d386 matmult
  ```

  You can then study any memory changes produced by your program using the memory display window. To see the contents of an area of memory type 1. The cursor will then be moved to memory window 1. Then type b followed by the start address of the area you want to examine. The memory contents should then be displayed and you can see how much of the product matrix your program is computing.

  The command to quit *d386* is **q**.

- Details about the **IBM-PC System BIOS** for performing I/O in 80386 are available on the 80386 web page given above. However for this lab exercise it will be enough to call the various output methods given at the end of the skeleton program, namely **output_int**, **output_tab**, **output_newline**. The pseudo-code shown earlier gives the required order of calls to these methods for printing a matrix, and the arithmetic operations required for performing matrix multiplication. You will need to calculate the address in memory of each particular matrix element by considering:

  - how many rows come before the row it belongs to;
  - how much space in memory each row takes up;
  - and how far along the element is, in the row it belongs to.

  Remember that the elements are 32-bit integers. You may assume the elements' values are small enough that no overflow occurs in 32-bit integer arithmetic during any of the additions or multiplications required in matrix multiplication.

- When you write your method calls you will need to remember that the return address pushed onto the stack is 16 bit.

- For conditional jumps you will need to use the form 'Jxx LONG Label' if the label is more than 127 bytes from the Jump instruction.

Testing Your Program

- We have included some simple test data at the end of the skeleton for you to use in testing your program. The labels "matrixA" and "matrixB" give the addresses of two simple input matrices, and the label "matrixC" gives the address of some space reserved for the product matrix. You are also encouraged to design your own test data.

Unassessed

- You could try checking for overflow in the matrix multiplication arithmetic. If overflow occurs the calculation can be abandoned and a suitable error message printed.

Submission

- You should run your **matmult** program on the test data given in the skeleton and save its output to a file called **m_output.txt**. You can do this with the command

  ```
  matmult > m_output.txt
  ```

  at the DOS prompt. Check the contents of **m_output.txt** are what you expect before submitting.

- Submit a copy of your files **matmult.asm** and **m_output.txt** by logging on to Linux and typing **submit 12** in the appropriate directory.

Assessment

- Your PPT may ask you to demonstrate your program but may prefer to mark it using your output file.

`matrix_print`	2
`matrix_mult`	3
`Design, style, readability`	5
`Total`	10