EEL308 Minor Exam 1 Semester 1302 Full marks: 20 7 February 2014
(Permitted references: Textbook (Patterson, Hennessy) and lecture notes)
Absolutely no Internet access, use of Wikipedia, etc. This and/or any other malpractice will straightaway lead to 'F' grade in the entire course.
Please begin your answer to each question on a new page.

Good luck! ☺

1. (**8 marks**) Ideal speedup of an application obtained as a result of parallelisation is equal to the number of processors. Real speedup however depends on the fraction of the application that can be parallelised (degree of parallelisation) and other overheads incurred during parallelisation, chiefly the cost of communication amongst processors. The table below shows certain parameters for three instruction classes A, B and C.

Instruction class	Instruction frequency (%)	CPI for uniprocessor	Degree of parallelisation possible for this instruction class (%)	Communication overhead per instruction (in clock cycles) (N is the number of processors)
A	20	1	0	0
B	40	3	80	0.5*N
C	40	5	100	0.5*N

   a. Considering **no communication overhead**, what is the maximum overall speedup achievable, given an infinite number of processors? [3]
   b. Under the same condition as (a), what would be the minimum number of processors required to achieve a speedup of at least 4? [2]
   c. Now, **considering communication overhead**, find out the maximum speedup achievable with an infinite number of processors. [3]

2. (**9 marks**) Let us say we have somehow loaded two **single-precision floating point** (following IEEE 754 floating point standard) numbers, which are in ARM **integer registers** r0 and r1. Write ARM assembly code to multiply these two FP numbers (using integer operations and integer registers, of course) and **store the result in r0**. You **do not** need to call a procedure (for this function) or save registers on stack unless you modify them in your program. Also, you **do not need to implement guard, round or sticky bits**; and you can assume that the **rounding step is not needed**. However, you need to **detect** overflow/underflow (and **just call** a procedure "SYSCALL"). You can use the ARM instruction for integer multiplication (MUL) directly, if needed.

3. (**3 marks**) In the following piece of x86 assembly code, the instruction byte **addresses** are shown along side in **hexadecimal** format. All other numbers are decimal.

```
0x1000    movw EAX, [EDI+100]
addr1     pop EDI
addr2     add EDI,; 20
addr3     call MYPROC
addr4     push EDI
```

   a. Find addr1, addr2, addr3 and addr4 in hexadecimal format. [2]
   b. If, before this piece of code, Memory[SP] contains 200 and EDI contains 100, what will be their values after this code is executed? [1]