Version AHG/3

EGT3
ENGINEERING TRIPOS PART IIB

Friday 6 May 2022 9.30 to 11.10

Module 4F14

COMPUTER SYSTEMS

*Answer not more than **two** questions.*

All questions carry the same number of marks.

*The **approximate** percentage of marks allocated to each part of a question is indicated in the right margin.*

*Write your candidate number **not** your name on the cover sheet.*

STATIONERY REQUIREMENTS
Single-sided script paper

SPECIAL REQUIREMENTS TO BE SUPPLIED FOR THIS EXAM
CUED approved calculator allowed
Engineering Data Book

10 minutes reading time is allowed for this paper at the start of the exam.

You may not start to read the questions printed on the subsequent pages of this question paper until instructed to do so.

You may not remove any stationery from the Examination Room.

1 (a) Explain what is meant by a *pipelined* datapath. If the latency of each pipe stage is T, apart from one stage which has a latency of kT, find the value of k that maximizes the speed-up compared with a non-pipelined datapath. State any assumptions you make. [40%]

(b) The following three segments of C++ code each calculate the sum of the `n` elements of the integer array `x`.

```
// Code A

int i, sum=0;

for (i=0; i<n; i++) {
  sum += x[i];
}
```

```
// Code B

int i, sum=0;

for (i=0; i<n; i+=2) {
  sum += x[i];
  sum += x[i+1];
}
```

```
// Code C

int i, sum=0, sum2=0;

for (i=0; i<n; i+=2) {
  sum += x[i];
  sum2 += x[i+1];
}
sum = sum + sum2;
```

After straightforward compilation (no compiler optimizations), the execution times on a modern processor for large `n` are: Code A 2.5 s; Code B 2.0 s; and Code C 1.5 s. Suggest plausible explanations for these execution times. [30%]

(c) The following extract of MIPS code is to be run on a 2-way superscalar pipeline with data forwarding.

```
add $8,$9,$10      # $8 loaded with $9+$10
addi $8,$8,4       # $8 loaded with $8+4
sw $8,A($0)        # $8 stored at address A
add $8,$11,$12     # $8 loaded with $11+$12
sw $8,B($0)        # $8 stored at address B
```

As written, it is difficult to fully exploit the superscalar architecture because of the repeated use of `$8`. By replacing `$8` with the otherwise unused register `$13` in some of the instructions, show how the superscalar pipeline can be better exploited. Speculate as to whether this sort of *register renaming* must be performed by the compiler, or whether it could be managed dynamically by the hardware. [30%]

2 (a) Explain why the latency of the arithmetic logic unit (ALU) is of paramount importance in computer hardware design. If the ALU were half as fast, estimate the impact on performance for a typical MIPS datapath (i) without and (ii) with pipelining. State any assumptions you make. [20%]

(b) Describe the operation of a three-level, 64-bit carry-lookahead adder. How many gate delays are required to produce the sums and the carry-out from the 64th bit? [30%]

(c) As an alternative to the three-level design, consider a single-level, 64-bit carry-lookahead adder. Such an adder would appear to require gates with a fan-in of 65, e.g.

```
c64 = g63 + p63.g62 + p63.p62.g61 .... + p63.p62.p61...p0.c0
```

However, a 16-input AND gate can be constructed from 4-input AND gates as follows

```
Z = (A.B.C.D).(E.F.G.H).(I.J.K.L).(M.N.O.P)
```

and four of these 16-input AND gates can be connected together in a similar manner to produce a 64-input AND gate, all using gates with a fan-in of 4. A similar arrangement works for OR gates.

(i) Estimate the time and space requirements of the single-level 64-bit carry-lookahead adder. Assume that most gates should have a fan-in $\leq$ 4, but it is acceptable for the occasional gate to have a fan-in of 5. [20%]

(ii) Compare the asymptotic time and space requirements of single-level and multi-level n-bit carry-lookahead adders. [20%]

(iii) Why are designs with higher space requirements less attractive? [10%]

3 (a) Explain why the inclusion of a cache, between the CPU and main memory, generally improves a computer's performance. [10%]

(b) Consider the following three caches.

(i) Direct-mapped, 4 words per block, 65536 blocks.

(ii) 4-way set-associative, 4 words per block, 4096 sets.

(iii) Fully associative, 4 words per block, 32768 blocks.

For each case, state the size of the cache and where (i.e. which byte of which word of which block or set) the byte at address 0xA0973C8E would be stored were it to be fetched into the cache. State also what the value of the tag would be. Assume 32-bit addresses and words. Express cache indices and tags in hexadecimal. [30%]

(c) Consider performing a matrix multiplication on a typical modern PC using the C++ code in Fig. 1. On the same axes, sketch curves of execution time against `matrixSize` for the cases `transpose_c = false` and `transpose_c = true`. Explain briefly the curves' general characteristics. [25%]

(d) The processor in the PC in (c) has a baseline clock cycles per instruction (CPI) count of 1.0 when all memory references hit in the primary cache, and a clock speed of 2 GHz. The main memory access time is 50 ns, including all the cache miss handling, and the cache miss rate per instruction is 5% (i.e. there is, on average, one cache miss per 20 instructions). How much faster would the computer be if we were to add a secondary cache (between the primary cache and main memory) that has a 5 ns access time for either a hit or a miss, and a miss rate per access of 50%? State any assumptions you make. [25%]

(e) At first sight, a secondary cache miss rate of 50% per access may seem surprisingly high. Explain why this is in fact perfectly plausible. [10%]

```
int a[matrixSize][matrixSize];
int b[matrixSize][matrixSize];
int c[matrixSize][matrixSize];
int ct[matrixSize][matrixSize];
bool transpose_c;

....

if (transpose_c) {
  for (i=0; i<matrixSize; i++)
    for (j=0; j<matrixSize; j++)
      ct[i][j] = c[j][i];
}

// Calculate a = b * c
for (i=0; i<matrixSize; i++)
  for (j=0; j<matrixSize; j++) {
    r = 0;
    for (k=0; k<matrixSize; k++)
      if (transpose_c) r += b[i][k] * ct[j][k];
      else r += b[i][k] * c[k][j];
    a[i][j] = r;
  }
```

Fig. 1

END OF PAPER

THIS PAGE IS BLANK

Part IIB 2022

Module 4F14: Computer Systems

Numerical Answers

1. (a) Assuming no stalls, maximum speed-up when $k = 1$

3. (b) (i) Size 1 MiB, index 0x73C8, tag 0xA09, block offset 3, byte offset 2
   (ii) Size 256 KiB, index 0x3C8, tag 0xA097, block offset 3, byte offset 2
   (iii) Size 512 KiB, no index, tag 0xA0973C8, block offset 3, byte offset 2

   (d) 1.5 times faster