Version AHG/3

EGT3
ENGINEERING TRIPOS PART IIB

Friday 3 May 2024 2 to 3.40

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) The table below shows the 32-bit **R-format** used for the MIPS instruction `add`:

op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

MIPS16e is an extension to the MIPS ISA, offering 16-bit *compressed* instructions. The table below shows the 16-bit **RRR-format** used for the equivalent MIPS16e add instruction:

op	rs	rt	rd	funct
5 bits	3 bits	3 bits	3 bits	2 bits

Taking the two add instructions as a representative case study, discuss the likely benefits and disadvantages of regular 32-bit RISC instructions and compressed 16-bit alternatives. [50%]

(b) Describe the design of 16-bit ripple-carry and carry-select adders. For both types of adder, state (with justification) the asymptotic time and space requirements. Compare these requirements with those of a carry-lookahead adder. [50%]

2 (a) Discuss briefly the relative advantages and disadvantages of direct-mapped and set-associative caches. [20%]

(b) Sketch a hardware schematic of a 64 KiB direct-mapped cache with four-word blocks. Design the cache for a byte addressable memory system with 32-bit addresses and 4-byte words. Annotate your sketch to show clearly the number of blocks in the cache, and precisely how the address is divided into a tag, an index, a block offset and a byte offset. [30%]

(c) The following segment of C++ code calculates the transpose of the `R` × `C` matrix `a`, storing the result in the `C` × `R` matrix `b`.

```
for (i=0; i<R; i++)
  for (j=0; j<C; j++)
    b[j][i] = a[i][j];
```

Each element of `a` and `b` is one byte. With `R` fixed at 1500, the execution time of the code segment is found to be 8 ms for `C` = 1500 and 2000 ms for `C` = 150000. How might these execution times be explained in terms of cache misses? [25%]

(d) Now consider the following segment of C++ code.

```
for (k=0; k<100; k++)
  for (i=0; i<1500; i++)
    for (j=k*1500; j<(k+1)*1500; j++)
      b[j][i] = a[i][j];
```

(i) What is the purpose of this code segment? [10%]

(ii) What, approximately, would you expect the execution time of this code segment to be? Justify your answer. [15%]

3 (a) Explain what is meant by a *pipelined* datapath. What are pipeline *hazards*? [20%]

(b) A data forwarding unit is to be added to the MIPS datapath in Fig. 1. Sketch the relevant part of the enhanced datapath, showing how the forwarding unit is connected to the other components. [20%]

(c) In the following segment of C++ code, `x` is an array of 4-byte integers.

```
for (i=0; i<512; i++)
   for (j=1; j<512; j++)
      x[512*i + j] = x[512*i + j-1];
```

A buggy compiler converts the C++ into the following MIPS code, which is to be executed on the enhanced datapath in (b). Any remaining data hazards, and all branch hazards, are resolved by stalling.

```
        addi $11,$0,512     # $11 loaded with 512
        addi $10,$0,xStart  # $10 loaded with xStart
        addi $8,$0,0        # $8 loaded with 0
loop1   addi $9,$0,4        # $9 loaded with 4
loop2   lw $12,0($10)       # $12 loaded with word at address $10
        sw $12,4($10)       # $12 stored at address $10+4
        addi $9,$9,4        # $9 incremented by 4
        addi $10,$10,4      # $10 incremented by 4
        bne $9,$11,loop2    # loop back unless $9 = 512
        addi $8,$8,4        # $8 incremented by 4
        bne $8,$11,loop1    # loop back unless $8 = 512
```

The compiler has assigned `$8` to represent `i`, `$9` to represent `j` and is using `$10` to index into `x`. `$11` holds the constant `512` and `$12` is used as temporary scratch space.

(i) There is a missing MIPS instruction immediately before the final `bne`. What should this instruction be, for correct compilation of the C++ code? [15%]

(ii) At which points in the program execution will the pipeline have to stall, and for how many clock cycles? [15%]

(iii) What performance improvements, if any, are achievable by: the compiler reordering the instructions; the compiler unrolling the inner loop; the programmer switching the order of the C++ loops? [30%]

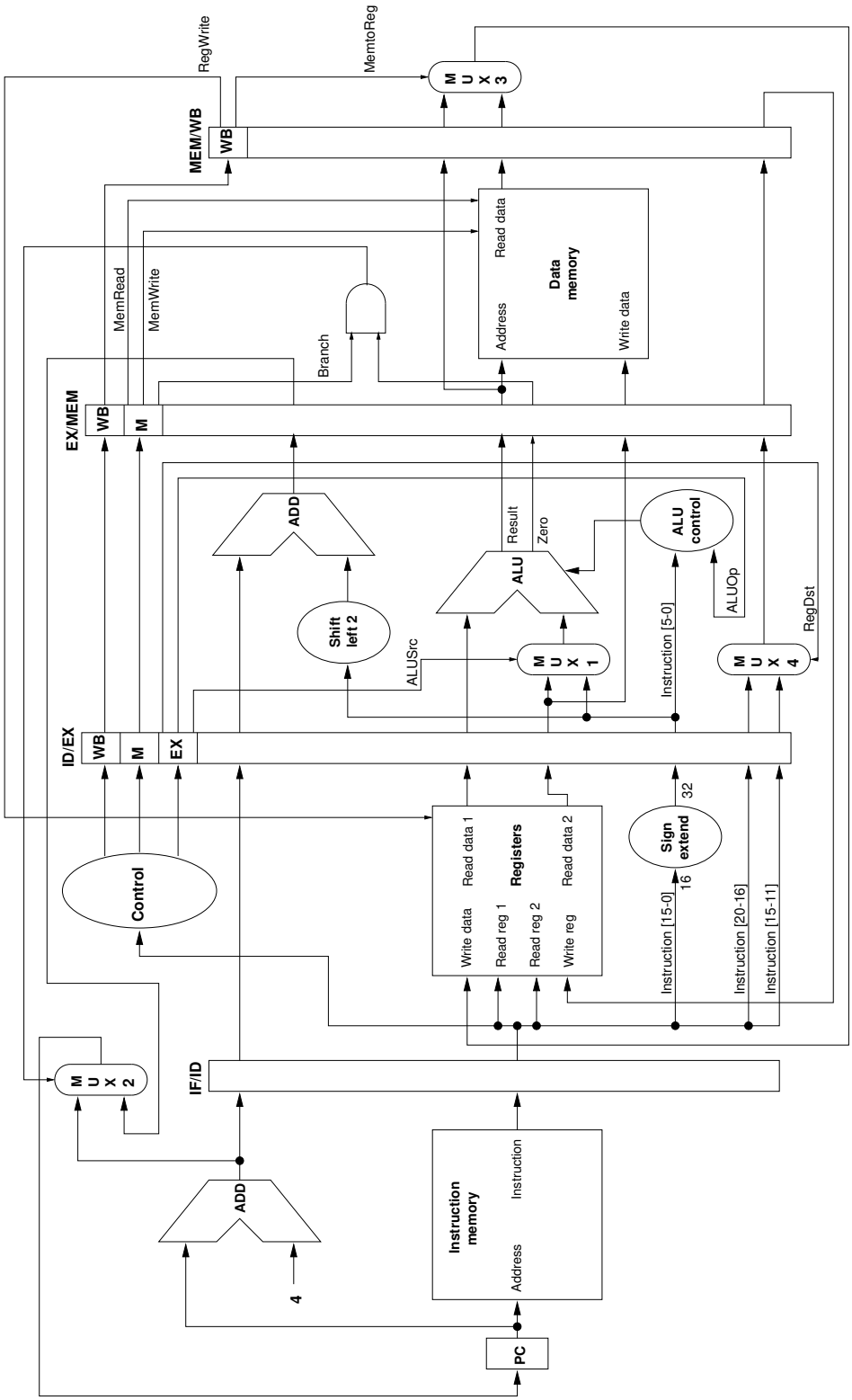

Fig. 1

END OF PAPER

THIS PAGE IS BLANK

Part IIB 2024

Module 4F14: Computer Systems

Numerical Answers

2. (d) (i) Transposes `a` into `b`
   (ii) ~ 800 ms