Endterm Computer Organisation CSE1400

Please read the following information carefully!

- You have 90 minutes to complete this exam.
- Before you hand in your answers, check that your multiple-choice form contains your name and student number, also filled in using the boxes.
- Opening this exam before you are instructed to start is **strictly prohibited**.
- The use of the book, notes, calculators, smart watches, and other aids is **strictly prohibited**.
- Note that the order of the letters next to the boxes on your multiple-choice sheet may **not always be A-B-C-D**!
- Fill in the answer form with **(dark) pencil** or **pen**. If you make a mistake on the answer form, you need to either erase the mistake or copy all answers to a new form.
- This exam consists of 18 multiple-choice questions. Each four-choice question is worth 250 points, each two-choice question is worth 125 points, for a total of 4000 points. The distribution of points over the questions is as follows:

Question:	1	2	3	4	5	6	7	8	9	10
Points:	250	250	250	250	250	250	250	250	250	250
Question:	11	12	13	14	15	16	17	18		Total
Points:	250	125	125	250	250	125	125	250		4000

1. (250 points) Consider an ISA with 64 bits per instruction. This ISA should support 42 registers and at most 32 MiB of byte-addressable main memory. 42 instructions (out of all the possible instructions) have two direct memory addresses and one register as operands. How many other instructions may still be created within the same ISA? Assume the number of bits used for an opcode is constant over all instructions.

   A. 22

   B. 64

   C. 86

   D. 214

   Solution: 42 registers, so 6 bits for each register operand. 32 MiB, so 25 bits per memory operand. Two memory operands and a register $= 25 \cdot 2 + 6 = 56$ bits taken, 8 left for the opcode. So $2^8 = 256$ instructions may be created in total. $256 - 42 = 214$ unique instructions can still be constructed.

   $22 = 64 - 42$ is not enough as this does not take into account the unused bits.

2. (250 points) A machine executes the 64-bit assembly code below, starting at the first line. The code uses AT&T syntax, so the order of the operands is "source, destination".

```
1  movq $string, %rdi
2  movq $5, %rsi
3  movq $0x0100, %rbx
4  movq $0, %r15
5  xorq %rax, %rax
6  call printf
7
8  movq $8, %rax
9  movb $0x50, %bl
10 addq %rbx, %rax
11 orq  %r15, %rax
```

   What will the content of the `%rax` register be after executing every instruction in this code? Assume `$string` is the address of some zero-terminated string.

   A. 0

   B. 88

   C. 344

   D. Unknown

   Solution:

   A. 0, if line 11 does an 'and' instead of an 'or', 0 will be the answer because R15 contains 0.

   B. 88, if line 9 were to reset the unused bits in RBX to 0, this will be the answer because RBX will then end up being 0x50.

C. 344, before line 10 RBX will be 0x150 $= 256 + 80 = 336$ and RAX will be 8, so new RAX will be

D. Unknown, because R15 and RBX are callee saved, we do not have to worry about their values getting lost through the call to printf.

3. (250 points) Which of the following statements is **correct**?

A. Synchronous busses have explicit handshaking.

B. A BPU with a word size of n has an I/O bus with n data lines and n address lines, but not necessarily also n control lines.

C. The correct interrupt service routine is found by the CPU in a table in memory and initialised by the CPU before boot.

D. An I/O device can connect to multiple I/O busses.

Solution:

A. explicit handshaking is Asynchronous busses –> false

B. A BPU does not need to use n data lines/addresses on a bus –> false

C. before boot? no... plus the OS does that

D. yes they can; nothing is stopping that. The CPU itself is a good example.

4. (250 points) Which of the following statements is **correct**?

A. DMA leads to higher throughput, but not to lower latency.

B. DMA controllers are one-trick ponies and should therefore be avoided.

C. Using DMA means that the CPU might not get enough time to handle other interrupts.

D. To prevent collisions on the bus, the CPU is the bus master during all DMA transfers.

Solution:

A. latency is not affected by DMA, throughput is

B. they do that one trick very well, do not avoid

C. nope; less interrupts –> more time for the others

D. DMA controllers can be bus master

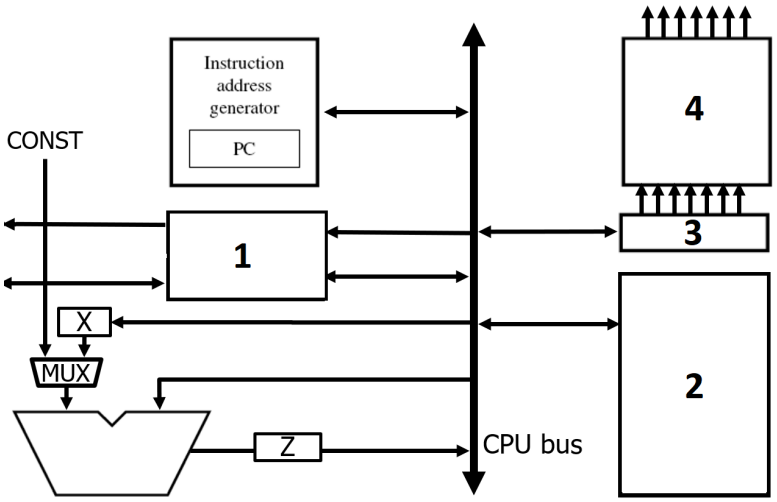

Figure 1: BPU Organisation

5. (250 points) Figure 1 shows an overview of the components in a basic processing unit. What are the elements labelled 1-4?

   A. 1. Processor-memory Interface 2. Register File 3. Instruction Register 4. Control Circuit

   B. 1. Processor-memory Interface 2. Control Circuit 3. Instruction Register 4. Register File

   C. 1. Control Circuit 2. Processor-memory Interface 3. Instruction Register 4. Register File

   D. None of the options listed is correct

6. (250 points) At the end of our current subroutine we want to return to the previous subroutine, `foo`. Before we can return we have to move the contents from *MDR* to *R0*. For convenience we have stored the return address for `foo` in R1. What is the fastest (but still correct) way of returning to `foo`?

A	B	C
$R1_{out}$, MAR_{in}, READ MDR_{out}, $R0_{in}$, WFMC MDR_{out}, PC_{in}	MDR_{out}, $R0_{in}$ $R1_{out}$, MAR_{in}, READ, WFMC MDR_{out}, IR_{in}	MDR_{out}, $R0_{in}$ $R1_{out}$, PC_{in}

   A. A

   B. B

   C. C

   D. None of the options is correct

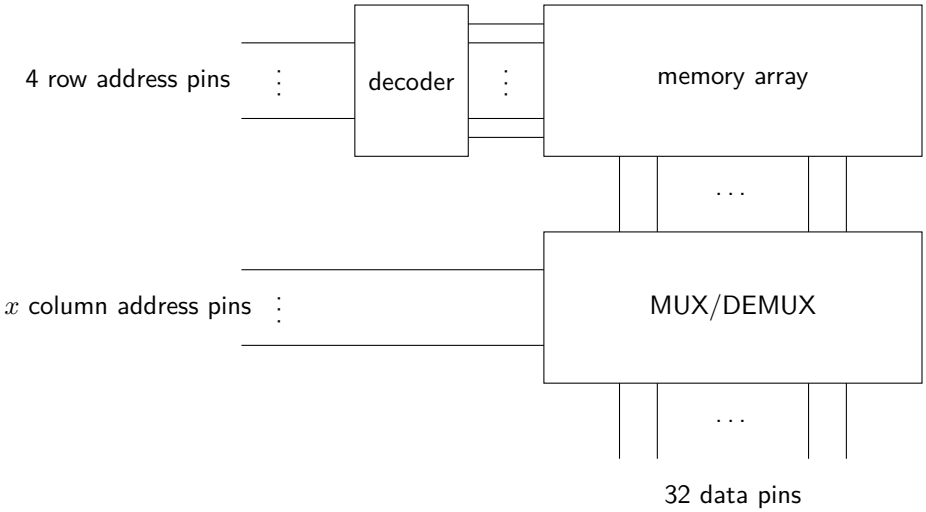

Figure 2: A memory module

7. (250 points) Figure 2 shows a memory module. The memory array can store 1024 bytes of data. How many column address pins are needed to access the data?

   A. 2 column address pins

   B. 4 column address pins

   C. 8 column address pins

   D. 16 column address pins

> **Solution:** $1024 = 2^{10}$ bytes $= 2^{13}$ bits. There are 4 row pins and 2^5 data pins, so there are 4 column pins needed $(4 + 5 + 4 = 13)$.

8. (250 points) Consider the code below, a fully-associative cache using a LRU replacement algorithm with 256 data blocks of 8 32-bit words, and assume that the variable `SUM` is stored in a register (as it is accessed frequently), while the numbers in array `A` (one word each) are located consecutively in memory.

```
1 int SUM = 0;
2 for (int j = 0; j < N; j++) {
3     SUM = SUM + A[j];
4 }
5 for (int j = 0; j < N; j++) {
6     A[j] = A[j] / SUM;
7 }
```

   For what value of `N` will *trashing* occur?

   A. 256

   B. 2048

   C. 4096

D. Trashing can only occur with direct-mapped caches

Solution: Each cache block holds 8 numbers (words), so in total the cache holds 256 $\times$ 8 $=$ 2048 numbers. Thus `N` needs to be larger than that for trashing to possibly occur. With `N`=4096, the numbers `A`[0-2047] are evicted (trashed) by numbers `A`[2048-4095] in the first loop, just before being needed by the second loop. Ouch.

9. (250 points) On a certain Thursday morning after a beach party, two of the Sharks decided to build a new coffee machine. It needs to be bi-lingual and should connect to the TUMIB (TU Massive I/O Bus) so they decided the I/O interface is going to be as follows:

   data lines: 20 Requirement of the bus; every message contains 840 bytes which for this machine contains the type of coffee wanted, the bean blend, amount of caffeine, amount and type of milk, amount of sugar, etc. (NB: pumpkin spice chai latte is not supported, that would require another 7 bytes).

   control lines: 8 Requirement of the bus; controls data transfers, signals coffee completion, etc.

   address lines: 20 Requirement of the bus

   To be bi-lingual the machine needs to respond to the following addresses:

   - `0xC0FEE`
   - `0xCAFEA`

   For yet unknown reasons [1] the Sharks decided to ignore the following address lines: 1-3, 6-9, 13, and 15.

   Which of the following (potentially unwanted) side effects is **not** happening?

   A. The machine responds to `0xC0C0A`

   B. The machine responds to `0xC0CE0`

   C. The machine responds to `0xC0CEA`

   D. The machine responds to `0xC0DE0`

Solution:

```
c0fee   1100 0000 1111 1110 1110
cafea   1100 1010 1111 1110 1010
common: 1100  0 0 1111 1110 1 10    so: 2, 13, 15

c0c0a   1100 0000 1100 0000 1010
diff                00 000          so: 5-9; 5 is not ignored

c0ce0   1100 0000 1100 1110 0000
diff                00      0 0     so: 1, 3, 8-9

c0cea   1100 0000 1100 1110 1010
diff                00              so: 8-9

c0de0   1100 0000 1101 1110 0000
diff                0       0 0     so: 1, 3, 9
```

[1] rumour has it that the Ducks gave țuică to the Sharks at the beach party

10. (250 points) A superscalar processor with a four-stage pipeline and two decode units has to execute the program written below. Each stage takes 1 cycle to complete. Reading an operand from memory causes a stall and therefore adds a 2 cycle delay.
Consider the program:

```
1 movq (%rax), %r9
2 movq $5, %r8
3 addq %r9, %r8
```

How many cycles does it take to execute the program?

A. 5

B. 6

C. 7

D. 8

Solution:

```
I1 FDDDEW
I2 FDEW
I3 F D  EW
```

Block	Counter ($c_3c_2c_1c_0$)
B0	0001
B1	0100
B2	0101
B3	0011
B4	0111
B5	0000
B6	0110
B7	0010

Table 1: Counter of each block B0-B7

11. (250 points) Consider a set-associative cache with 8 blocks per set labelled B0 to B7. The cache uses a LRU replacement algorithm. The initial state of the counter for each block is shown in Table 1. Assume the following events occur:

    1. A cache hit occurs for B5
    2. A cache miss occurs twice
    3. A cache hit occurs for B2

    What block will get replaced if another cache miss occurs?

    A. B1

    B. B2

    C. B3

    D. B6

Solution:

Block	Initial	B5 hit	1st miss	2nd miss	B2 hit
B0	1	1	2	3	4
B1	4	4	5	6	7
B2	5	5	6	7	0
B3	3	3	4	5	6
B4	7	7	0	1	2
B5	0	0	1	2	3
B6	6	6	7	0	1
B7	2	2	3	4	5

After the last hit, B1 has the highest counter and will be replaced. If it is assumed LRU counts down instead of up, the answer would be B3.

12. (125 points) True or false:

    Pipelining boosts CPU performance by reducing the time to execute a single instruction.

    A. True

    B. False

Solution: Pipelining uses parallelism to increase the throughput; the latency stays the same.

13. (125 points) True or false:

    The interstage buffer between the decode and execution stage in a pipelined RISC processor includes the destination register in case of arithmetic assembly instructions (ADD, SUB, etc).

    A. True

    B. False

Solution: The destination register is needed further down the pipeline by the write-back stage.

14. (250 points) The Ducks and Sharks continue work on the coffee machine of question 9. They are building the coffee machine with a byte-addressable memory system with 2 MiB of memory that makes use of a set-associative cache of 64 KiB. The Ducks want to make sure that the cache blocks are 256 bytes in size. The Sharks, on the other hand, wants the tag part of the memory address to be exactly eight bits.

    For this n-way associative cache, what is the value of n?

    A. 4

    B. 8

    C. 16

    D. 32

Solution: The memory addresses consist of a tag part, a set part, and a word part, totalling 21 bits (because there are 2^{21} bytes of memory).

256 bytes per block, so 8 bits are used for the word part.
Tag part is fixed to 8 bits.
The other 5 bits can be used for the set part, which means that the cache contains 32 sets.

The cache has 2^{16} bytes, or 2^8 blocks.
Dividing these blocks over 32 sets gives 8 blocks per set, thus it's 8-way associative.

15. (250 points) When designing an OS supporting virtual memory, the size of the page table is an important design parameter. Greg does not want to waste memory space, so he insists on keeping the page table size below 12.5 % of the available physical memory. Fred argues that, as a consequence, the size of the virtual address space must then be limited.

    What will be the size of the virtual address space for a 64-bit processor equipped with 32 GiB of main memory and using a 16 KiB pages when Greg gets his way? Assume that a page table entry is byte-aligned and contains, next to the mapping info, 5 control bits.

    A. 8 TiB

    B. 12 TiB

C. 16 TiB

D. 21 TiB

Solution: The pagetable size must be 12.5% of 32 GiB = 4 GiB. One entry holds a physical page number and 5 control bits. There will be 32 GiB / 16 KiB = 2M pages in main memory, which requires 21 bits for the physical page number. Making it byte aligned the 26 bits will be stored in 4 bytes (what a waste; don't tell Greg!). That means the pagetable can hold 4 GiB / 4B = 1G entries, effectively addressing 1G × 16 KiB = 16 TiB of virtual memory.

16. (125 points) True or false:

GPUs (graphical processing units) belong to the MIMD category according to the Flynn taxonomy.

A. True

B. False

Solution: GPUs are Single-Instruction-Multiple-Data (SIMD), not MIMD.

17. (125 points) True or false:

A single-core CPU that uses pipelining belongs to the SISD category according to the Flynn taxonomy.

A. True

B. False

Solution: Even though the CPU uses pipelining, it is still Single-Instruction-Single-Data.

18. (250 points) Aad created a program to draw an image of a Mandelbrot set in a small terminal. He wants to speed up his program, as it takes his original version about 30 seconds to draw the image. After making his program massively parallel, he runs it again on a GPU with 201 cores. His program now completes in 20 seconds. How much is the fraction f_p of the program he managed to parallellise? Assume his GPU has the same speed as his CPU and assume changing from a CPU to GPU introduces no overhead.

A. $f_p = \frac{66}{201}$

B. $f_p = \frac{67}{200}$

C. $f_p = \frac{122}{200}$

D. $f_p = \frac{66}{200}$

Solution: The program runs in 30 seconds with just 1 core. It runs in 20 seconds with 201 cores.

We can fill in Amdahl's law to find fraction f_p:

$$\begin{aligned} T_p &= T_s \cdot (f_s + \frac{f_p}{p}) \\ 20 &= 30 \cdot (f_s + \frac{f_p}{201}) \\ \frac{2}{3} &= f_s + \frac{f_p}{201} \\ \frac{2}{3} &= (1 - f_p) + \frac{f_p}{201} \\ \frac{200}{201} f_p = \frac{1}{3} f_p &= \frac{1}{3} \cdot \frac{201}{200} = \frac{67}{200} \end{aligned}$$

Where the substitution $f_s = 1 - f_p$ comes from $f_s + f_p = 1$.

Alternatively, one may fill in and solve a system of equations defined as $\frac{a}{p} + b = T_p$ where a and b are constants, p is the number of processors used, and T_p is the time associated to the number of processors used. We solve a for $\frac{a}{1} + b = 30$ and $\frac{a}{201} + b = 20$ by solving the linear system of equations.

$$\begin{aligned} a + b &= 30 \\ \frac{a}{201} + b &= 20 \\ &\text{So:} \\ \frac{200}{201} a &= 10 \\ a = \frac{2010}{200} &= \frac{201}{20} \end{aligned}$$

We now need to find what fraction a is of the length of the entire program to find f_p. We do $f_p = \frac{a}{T_1} = \frac{\frac{201}{20}}{30} = \frac{201}{600} = \frac{67}{200}$.

End of exam