Part I: Fundamentals

1. Module 1: C and Assembly Programming

(a) Short answer: Register $t1$ is modified and contains value $48_{10}$ after that the instruction is executed.

Elaborated answer: The first step is to decode the instruction. The 6 most significant bits are all zero, which means that the op-field is zero. Hence, we know that it is an R-type instruction. The funct field contains value $4_{10}$. If we look at the MIPS reference sheet, we see that the instruction must be sllv (shift left logic variable). By decoding the complete instruction, we get:

```
sllv $t1,$t0,$s1
```

Consequently, we get value 6 from $t0$, shift it 3 bits to the left (value read from $s1$). Shifting 3 bits left is the same as multiplying by 8. Hence, we get the end result $48_{10}$, which is written to register $t1$.

Max 4 points. Two points if register $t1$ is given and two points if the resulting value is $48_{10}$.

(b) Short answer: 9.9 32

Elaborated answer: First, go through the for-loop and compute the updated values in array v. The answer v[1] picks the second element in the array (remember that a C array starts at index 0). Hence, you do not need to follow the loop more than two iterations. What happens is that the first element is updated to 4.5 because it is added together with value 2 from a (using pointer p2). The second element is updated by adding together the new value in a, which is 4.5, together with the original value in the second element in v (value 5.4). Adding these numbers together gives 9.9, which is stored in the second element of v.

As was given in the exercise, a double precision floating point number is 64 bits, which is the same as 8 bytes. Hence, the size of the array is 8 times 4, that is, 32 bytes.

Max 4 points. Two points for the floating point number and two points for the size in bytes. We give 1 point for the answer 4.5 for the floating point number (correct pointer arithmetic, but incorrect array indexing).
2. Module 2: I/O Systems

(a) Answer: A possible solution:

```assembly
led4:  # label 'led4' states the function name
    lui  $t0,0xbf22  # load the whole 32-bit address
    ori  $t0,$t0,0x6000
    andi $a0,$a0,1  # mask so that we only use bit 0
    sll  $a0,$a0,3   # shift the value to LED 4 position
    lw   $t1,0($t0)  # load the previous setting
    andi $t1,$t1,0xfff7 # clear bit 3
    or   $a0,$t1,$a0  # insert the bit for the led
    sw   $a0,0($t0)   # write to the memory mapped IO
    jr   $ra  # return from function call
```

Max 5 points. 1 point if the function includes both a label and a return jump (jr). 1 point if the address is loaded correctly (using lui and ori). 1 point if the argument is masked correctly (using andi) and shifted correctly (using sll). 1 point if the current port value is loaded (using lw), together with correct masking (using e.g., andi and or). Finally, 1 point if the value is stored correctly (using sw).

(b) Answer:

i. True.

ii. False. DMA is used locally by the processor to transfer data between different memory locations.

iii. True.

Max 3 points. One point for each correct answer. A false statement needs to include a correct motivation.
3. **Module 3: Logic Design (for IS1500 only)**

(a) Short answer: \( B\bar{C} \)

Elaborated answer: The solution can be obtained by first using the sum-of-products form, and then perform simplifications. \( \bar{A}B\bar{C} + AB\bar{C} = B\bar{C}\bar{A} + B\bar{C}A = B\bar{C}(\bar{A} + A) = BC. \)

Max 2 points. 2 points for a correct solution. Answers that are not written in simplified form are also given 2 points.

(b) Short answer: The value is \(-13\).

Elaborated answer: Convert the hexadecimal value \(0xF3\) to a binary value \(11110011\). Perform the two’s complement algorithm: invert all bits and add value 1. This gives the binary value \(00001101\). Hence, the decimal interpretation is \(-13\).

Max 2 points. 2 points if exactly the correct answer, else 0 points.

(c) Answer:

<table>
<thead>
<tr>
<th>( A_1 )</th>
<th>( A_0 )</th>
<th>( Y_1 )</th>
<th>( Y_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Max 4 points. 1 point for each correct row.
4. Module 4: Processor Design

(a) Short answer: $A = 0xa$, $B = 0x1200ff00$, $C = 0x1000800a$, and $D = 0x00400004$.

Elaborated answer:

Signal $A$: The $sw$ instruction is an I-type instruction. Signal $A$ is the bits 20:16 of the instruction, which is the $rt$ field of an I-type instruction. For the $sw$ instruction, the $rt$ field is the register that contains that data that should be stored, in this case $s2$. Register $s2$ is encoded with number 10 in decimal form. This means that $A = 0xa$.

Signal $B$ is the data that should be written to the data memory. We know that the data is stored in register $s2$, which means that $B = 0x1200ff00$.

Signal $C$ is the result from the ALU. For the store word instruction, the $ALUSrc$ control signal is 1 because the store word instruction adds together the base address (the value stored in $s0$) and the index value (in this case value $-8$). Hence, $C = 0x1000800a$.

Signal $D$ is a branch address, if this instruction would have been a relative branch instruction. As always in digital logic, the signal value is decoded, even if it is not used in this case (the control unit determines that this is not a branch instruction). Signal $D$ is computed by taking the current PC value, adding value 4, and then adding the immediate value after left shifting by two. Hence, $D = 0x00400004$.

Max 4 points. One point for each correct signal value.

(b) Answer:

- It is only the $beq$ instruction that can cause a hazard.
- This instruction causes a control hazard, if the branch is taken.
- The hazard is handled by flushing the pipeline and fetching a new instruction.

We can also note that that there is a write after read (WAR) hazard. Note, however, that such a hazard is only relevant if this is an out-of-order processor that can change the order of how the instructions are executed.

Max 4 points. Two points if only $beq$ is given and one point if $beq$ is given together with some other instruction. One point if it is clear that is only a control hazard. One point if it is clear the hazard is handled by flushing. Give 1 point for stating that there is a WAR hazard, which is true, even if it is not relevant for this 5 stage pipeline.
5. **Module 5: Memory Hierarchy**

(a) Short answer: The tag field is 7 bits, the set field is 6 bits, and the byte offset field is 3 bits. There are 256 valid bits in the cache.

Elaborated answer:

There are $2048/4 = 512$ bytes capacity in each way. Since there are 8 bytes in each block, the byte offset field is 3 bits. Also, we have $512/8 = 64$ sets. This means that the size of the set field is 6 bits because $2^6 = 64$. Finally, since we have a 16-bit address, the tag field is $16 - 6 - 3 = 7$ bits. Since there are 64 sets and 4 ways, there are 256 valid bits.

Max 4 points. One point for each correct answer.

(b) Short answer: The data cache can neither utilize\(^1\) temporal nor spatial locality and the data cache miss rate is 100%. The instruction cache utilizes both spatial and temporal locality and the instruction cache miss rate is $\frac{3}{403}$.

Elaborated answer:

The only instruction that uses the data cache is `lw`. It starts to load at address 0x10010190 ($400_{10} = 0x190$), which means that it starts at the first word of a cache block. However, since the cache block is 8 bytes large and it will move $s0$ 8 bytes in each iteration, it will never get a cache hit. Hence, the cache miss rate is 100%. The data cache can neither utilize temporal nor spatial locality. If the block size would have been larger, the data cache could have utilized spatial locality due to the 8-byte increments.

The first `addi` instruction is located at address 0x0040100c, which is the last word in a cache block. This means that there will be one miss at the first `addi` instruction. The next miss will be at the `lui` instruction, followed by a miss at the next `addi` instruction (within the loop). After that, there will be no more cache misses. There are in total $3 + 4 \times 100 = 403$ memory accesses. Hence, the cache miss rate for the instruction cache is $\frac{3}{403}$. The instruction cache utilizes both spatial and temporal locality.

Max 4 points. One point for each correct answer about cache hit rate. One point for each of the caches if the answer about temporal and spatial locality is correct. We accept correct answers for both miss rate and hit rate.

\(^1\)2017-03-14: Clarified the solution about utilizing spatial locality.
Module 6: Parallel Processors and Programs

(a) Answers:
   i. False. AVX is a standard for implementing SIMD or subword parallelism.
   ii. False. Moore’s law basically says that the number of transistors on an integrated circuit doubles every 18-24 months.
   iii. True.
   iv. False. A multicore processor utilizes task-level parallelism. Each core has a different instruction stream (not the same instruction stream as i data-level parallelism).

Max 4 points. One point for each correct answer. A false statement needs to include a correct motivation.

(b) Short answer: The relative speedup needs at least 7 cores, whereas the true speedup needs 32 cores.
Elaborated answer:
To calculate the fraction between what can be parallelized and what cannot, we can use Amdahl’s law. If \( x \) is the number of cores, 9 is the maximal speedup, and \( y \) is the time that is unaffected by parallelization, we have \( \frac{9}{\frac{x}{x+y}} = 9 \). If we let \( x \) go to the infinity, we see that \( y = 1 \). That is, 1 second of the execution time is not affected of parallelization.

Relative speedup: Again, using Amdahl’s law, we have \( \frac{9}{\frac{5}{5+1}} = 4 \), which gives \( x = \frac{32}{5} = 6.4 \). This means that we need at least 7 cores to get a relative speedup of 4.

True speedup: We need to compare with the fastest available sequential implementation. This means that we replace the numerator with the value of the other persons implementation, that is, 5 seconds. We then have \( \frac{5}{\frac{5}{5+1}} = 4 \), which gives \( x = 32 \).
Hence, we need at least 32 cores to achieve a real speedup of 4.

Max 4 points. Two points for each speedup number. 1 point for a number that is correct, but where the term true and relative is switched. 1 point for answering \( \frac{32}{5} \) or 6.4 instead of the integer number 7.
Part II: Advanced

7. Dear Newspaper Editor,
Recently, I read an advertisement in your newspaper that included a number of incorrect statements and some direct lies. The advertisement was created by a company called “F-Aked Electronics AB” and concerned their new so called computer. The title of the advertisement was “The fastest computer on earth is here!”. To make it clear how incorrect this advertisement was, I will now summarize the main issues with the text:

- The advertisement says that their processor can run at higher clock frequency than 10 GHz. But, this is impossible! The processor would be way too hot. They say that the processor would give faster computation, but since it is not possible to have this high clock frequency, this must be a lie too. Moreover, they claim that this processor uses just half the amount of energy. This also sounds unlikely, because higher clock frequencies usually requires high voltage. Processor manufactures usually tries to lower the clock frequency to save energy, indirectly by also lowering the voltage.

- They claim that they use a new technology called simultaneous multithreading (SMT) with certain capabilities of combining hardware multithreading and multiple issue. But, this is not new! It has been used in commercial processors for many years. For instance, when Intel is using SMT, they call it hyper-threading.

- They claim that their product does not need to use a cache coherence protocol. But then they describe all the normal configurations that need a cache coherence protocol: multilevel caches (L1 and L2) on a multicore processor. It sounds unlikely that they can do this without a cache coherence protocol since they also claim that the processor is a shared memory processor (SMP). However, we cannot say if they are wrong or right about this, but I am skeptical.

- Then they say something about hardware support for MapReduce. This statement is not understandable at all. They say that this avoids the need to declare semaphores to get accurate timers. What does this mean? Semaphores and timers are orthogonal concepts, so is MapReduce. MapReduce is a programming model that is very good for big data computations, for performing batch processing on large cluster systems, whereas semaphores are used for handling shared resources correctly in concurrent programs. This is not just misleading but completely wrong!

- Their last claim is that they only use DRAM memories and no flash memories. This might be the case, but their rationale for why they do it is actually incorrect. In contrast to what they say, DRAM memories need to be periodically refreshed, and the data is lost when the power is turned off. However, flash memories are not volatile and the data is preserved if the power is turned off.

Clearly, this advertisement is really misleading and should not have been published. I understand that is very hard (if not impossible) for you as the editor to check all texts, but something must be done in this case. I hope that you will take appropriate actions against this company and also publish a correction about this in a future issue of the newspaper.

Best regards, David

Max 15 points. Max 3 points for each bullet item.
8. A possible solution is as follows:

```c
void mmul(const double* A, const double* B, double* C, int n){
    for(int i=0; i < n; i++){
        for(int j=0; j < n; j++){
            double t = 0;
            for(int k=0; k < n; k++){
                t += *(A + i*n + k) * *(B + j+k*n);
            }
            *(C + i*n + j) = t;
        }
    }
}
```

Max 15 points. We assume that the function is correct, and then remove points for each error. We do not remove points for syntax errors that do not change the meaning of the program. However, if the syntax error changes the meaning of the program, it is counted as an error.

9. Below, for each of the four concepts, you will find short explanations of what is expected from an answer. We do not provide any actual solutions here, because these explanations can be written in very different ways, with different examples. Instead, we give points to where you can find more information about the different topics.

- False sharing. Explain the main idea by using e.g., an example where two variables share the same cache block and where one variable is read by one thread and the other variable is written by another thread. For more information, see section 5.5 in Patterson and Hennessy and Lecture 13.

- Data hazards in a pipelined processor: Give an example that demonstrates both where forwarding is possible (e.g. in a simple dependency between two R-type instructions) and one example where we need to do stalling (e.g., in a load word instruction). See Lecture 10.

- Reading from an N-way set associative cache: The roles of the tag, the set, and the byte offset must be clear. The example should give a step-by-step guide, explaining how the set is selected, how the tag of the address is compared in the memory, and how the valid bit is checked. It should also mention that the ways are checked in parallel. An example can for instance be a 2-way or a 4-way set associative cache. See Lecture 11.

- Function calls in MIPS assembly: Explain how the stack is used for storing arguments that do not fit in registers and how it is used for local variables. Explain the idea of caller saved and callee saved registers. To get full point, the description needs to discuss nested calls, and how the registers need to be pushed on the stack. See Lecture 3 and Harris & Harris Section 6.4.6.

Max 20 points. Max 5 points on each item. To get high points on the is exercise, the text must be very clear and convincing and the examples must be clear and easy to understand.