Written Exam / Tentamen

Computer Organization and Components / Datorteknik och komponenter (IS1500), 9 hp
Computer Hardware Engineering / Datorteknik, grundkurs (IS1200), 7.5 hp

KTH Royal Institute of Technology

2020-06-05

14.00-19.00

Suggested Solutions

Part I: Fundamentals

In part I, on the real exam, only short answers are expected. The elaborated answers given here are just for your information, and are not needed on the real exam.

1. **Module 1: C and Assembly Programming**

   (a) Short answer:

   Statement #1:

   ```
   *foo = *foo * v;
   ```

   Statement #2:

   ```
   foo++;
   ```

   Max 4 points. 2 points for each correct statement. 1 point for a statement with only a minor error.

   (b) Short answer: `0x1560fffd`

   Elaborated answer: The `bne` instruction is an I-type instruction. The immediate field encodes the relative address that the branch instruction should jump to. According to the MIPS reference sheet, the branch target address is computed as follows.

   ```
   BTA = PC + 4 + signext(imm) * 4
   ```

   If we assume that the `bne` instruction is located at address `X`, then the `BTA` must be equal to `X - 8`. Hence, we have the equation

   ```
   X - 8 = X + 4 + signext(imm) * 4
   ```

   which is the same as

   ```
   signext(imm) = -3
   ```

   Consequently, by computing the two's complement number of `-3` and sign extending it to a 16-bit hexadecimal number, we get `0xfffd`.

   Max 4 points. 1 point for each byte of the machine code that is correct.

2. **Module 2: I/O Systems**

   (a) Short answer: One possible solution is to set the period register to $50\,000$ and to use the prescale value `1:4`.

   Elaborated answer: There are many possible solutions to this exercise. For instance, if we set the period register to $50\,000$, it means that we will trigger the interrupt $20\,000\,000/50\,000 = 400$ times every second. If we then set the prescale value to be `1:4` it will be triggered 100 times every second, which means that it will be triggered every $1/100 = 0.01$s, which is the same as every 10ms.

   Max 3 points. 3 points if both the prescale and the period values are correct. Note that there are several correct solutions, but the two values need to be consistent to give full points. Give 1 or 2 points if the solution is partially correct.

   (b) Short answer:

```
lui     $t2,0x1dbb
ori     $t2,$t2,0x8050
lui     $t1,0x2243
ori     $t1,$t1,0x0030

lw      $t0,0($t1)
srl     $t0,$t0,4
andi    $t0,$t0,0x3

lw      $t3,0($t2)
andi    $t3,$t3,0xff9f
sll     $t0,$t0,3
or      $t3,$t3,$t0
sw      $t3,0($t2)
```

   Max 5 points. One point for setting up the two registers with correct addresses. One point for correctly loading the switches values, shifting to the right, and masking (using `andi`). One point for loading the value for the LEDs and masking. One point for `or`-ing. One point for storing back the value to the LEDs port.

3. **Module 3: Logic Design (for IS1500 only)**

   (a) Short answer: $Y_0 = 0$, $Y_1 = 1$, $Y_2 = 0$, and $N = 2$.

   Max 2 points. 2 points if all values are correct. 1 point if at least 2 values are correct.

   (b) Short answer: 4096 bytes.

   Each register value has the size of 4 bytes (32 bits). We have in total $2^{10} = 1024$ such values. Hence, the register file can hold $4 \cdot 1024 = 4096$ bytes in total.

   Max 2 points. Two points for the correct answer. One point if the number is correct, but the unit is incorrect.

   (c) Short answers:

   - 32 bits
   - `unknown`
   - 1
   - 20

   Max 4 points. 1 point for each correct answer.

4. **Module 4: Processor Design**

   (a) Short answer: The signal values are $A = 24$, $B = \texttt{0x003f0011}$, $C = \texttt{unknown}$, $D = \texttt{0x40004060}$, $E = \texttt{0x4000404C}$

   Max 4 points. One point for each correct answer.

   (b) Short answer: Zero control hazards and zero data hazards.

   Comment: Note that the branch is never taken (no control hazard) and that register `$t1` value is already available in the pipeline when the `beq` instruction reads the register value of `$t1`. Hence, no data hazards either.

   Max 3 points. 1 point for answering zero control hazards and 1 point for answering zero data hazards. If answering both zero control hazards and zero data hazards, then 3 points.

5. **Module 5: Memory Hierarchy**

   (a) Short answer: The tag field is 22 bits, the set field is 6 bits, and the byte offset field is 4 bits. There are 256 valid bits in the cache.

   Elaborated answer:

   There are $4096/4 = 1024$ bytes capacity in each way. Because there are 16 bytes in each block, the byte offset field is 4 bits. Also, we have $1024/16 = 64$ sets. This means that the size of the set field is 6 bits because $2^6 = 64$. Finally, since we have a 32-bit address, the tag field is $32 - 6 - 4 = 22$ bits. Because there are 64 sets and 4 ways, there are 256 valid bits.

   Max 4 points. One point for each correct answer.

   (b) Short answers: i) $\frac{1}{3}$, ii) spatial locality, and iii) both temporal and spatial locality.

   Elaborated answer:

   i. The loop loops 6 times. Hence, there are in total 6 memory access to the data memory via the `lw` instruction. The data block size is 16 bytes and the first memory access is at address `0xa0201f28` (note the index address at the `lw` instruction). Since the memory address is incremented by 4 in each iteration, we get the following sequence: 1. cache miss, 2. cache hit, 3. cache miss, 4. cache hit, 5. cache hit, 6. cache hit. Hence, we have 2 cache misses and 6 memory accesses. We get a data cache miss rate of $\frac{2}{6} = \frac{1}{3}$.
   ii. The data instructions never read from the same memory address. Every second memory access, we get a hit. Hence, it uses spatial locality but not temporal locality.
   iii. In the instruction cache case, we can observe both temporal and spatial locality. We always read 4 instructions, which are used when executing several instructions in a row. We can also see temporal locality, since we are executing the same loop 5 times.

   Max 4 points. In the first question, we give two points for a correct answer and zero points if not correct. For the two last questions: one point each for a correct answer.

6. **Module 6: Parallel Processors and Programs**

(a) Short answer: The speedup would be $\frac{320}{41}$.

Elaborated answer: Let T be the time when the program executes on one core, and let p be the proportion of the time that is sequential, which cannot be parallelized. Using Amdahl's law, we have for the 6-core machine with a speedup of 4 the following equation:

$$4 = \frac{T}{\frac{T(1-p)}{6} + T \cdot p} \tag{1}$$

If we simplify the right hand side by getting rid of T, and solve the equation for p, we get $p = \frac{1}{10}$. We can now use the law again, and compute the speedup for 32 cores:

$$\frac{T}{\frac{T(1-\frac{1}{10})}{32} + T\frac{1}{10}} = \frac{1}{\frac{1-\frac{1}{10}}{32} + \frac{1}{10}} = \frac{320}{41} \tag{2}$$

Hence, we estimate that the speedup will be $\frac{320}{41}$ (approximately 7.8) when the program is executed on 32 cores.

Max 3 points. Three points for writing the correct fractional number. Two points if approximately the correct value (value 7, plus/minus 1). 1 point if Amdahl's law is setup in a correct way.

(b) Short answer:

i. False. Register renaming is used in out-of-order processors.

ii. True.

iii. True.

iv. False. Software treads are used in an operating system to enable concurrency and is not related to SIMD instructions.

v. True.

Max 5 points. One point for each correct answer, together with relevant explanations for the false answers.

Part II: Advanced

7. The complete solution is omitted. Please see the lecture slides and the course book.

   The question is graded in three levels S, G, and VG, with the following criteria:

   - Satisfactory (S): Some of the concepts in the question are clearly explained.
   - Good (G): Basically all concepts in the question are clearly explained, and some of the concepts are related to each other, by discussing similarities and differences.
   - Very Good (VG): Basically all concepts in the question are clearly explained, and all of the concepts are related to each other, by discussing similarities and differences.

   If none of the three levels is achieved, the exercise is considered as failed (F).

8. The following C code shows an example of a C code solution.

```c
#include <stdio.h>

const char* text1 = "This␣is␣a␣text.";
const char* text2 = "Three␣lines␣\n␣and␣\n␣␣numbers␣8812␣121.";

int count(const char* p, int* chars_p){
  char c;
  int words = 0;
  int in_space = 1;
  int chars = 0;
  while(*p){
    c = *p;
    p++;
    if(c == '␣' ||  c == '\t' || c == '\n')
      in_space = 1;
    else{
      chars++;
      words += in_space;
      in_space = 0;
    }
  }
  *chars_p = chars;
  return words;
}


int main(){
  int chars;
  int total_chars = 0;
  printf("Words␣in␣text1:␣%d\n", count(text1,&chars));
  total_chars += chars;

  printf("Words␣in␣text2:␣%d\n", count(text2,&chars));
  total_chars += chars;

  printf("Total␣chars:␣%d\n",total_chars);
  return 0;
}
```

The question is graded in three levels S, G, and VG, with the following criteria:

- Satisfactory (S): Some minor parts of a C or Assembly program are constructed correctly, but there are major errors or missing parts of the program.
- Good (G): The majority of the program is constructed correctly, including the main flow and structure of the program, but there are a number of minor errors within the program.
- Very Good (VG): The program is correct, with basically no errors.

If none of the three levels is achieved, the exercise is considered as failed (F).

9. Potential solutions for the three different sub-problems L1, L2, and L3 are given below.

   - L1: After `jal foo` (third instruction) returns, execution will fall through again to `foo`, this time with `$a0` $=$ `&numbers`$+40$ and `$a1 = 0`. In the first iteration, the loop condition will check if $-1 = 0$, in the second if $-2 = 0$, and so on. Therefore, the loop never terminates and writes the integers $0, -1, -2, -3, \ldots$ to the addresses `&numbers` $+ 40$, `&numbers` $+ 44$, `&numbers` $+ 48, \ldots$.
   - L2: In the following, we assume that `la` is a real instruction, because nothing was stated about this in the exercise.
     - $N = 2$
     - $C = 4096 = 2^{12}$
     - $S = 128 = 2^7$ implies $b = C/(S * N) = 2^12/(2^7 * 2^1) = 2^4 = 16$. That is, block size is 16 bytes, or 4 instructions.
     - `&foo` $=$ `0x40001a0c` implies that `foo` starts at fourth and last instruction in block.
     - Entire program fits in cache (using only 1 way).

     In the below, $O =$ `hit`, $X =$ `miss`

```
start: la   $a0, numbers  # Block instr. 1: | X
       addi $a1, $0, 10   # Block instr. 2: | O
       jal  foo           # Block instr. 3: | O

foo:   addi $sp, $sp,-4   # Block instr. 4: | O  O
       sw   $a0, 0($sp)   # Block instr. 1: | X  O

                          # *10  Forever...
loop:  sw   $a1, 0($a0)   # Block instr. 2: | OO...0 00...
       addi $a0, $a0, 4   # Block instr. 3: | OO...O OO...
       addi $a1, $a1, -1  # Block instr. 4: | OO...O OO...
       bne  $a1, $0, loop # Block instr. 1: | XO...O OO...
       lw   $t3, 0($sp)   # Block instr. 2: | O
       addi $sp, $sp, 4   # Block instr. 3: | O
       lw   $t0, 0($t3)   # Block instr. 4: | O
       lw   $t1, 4($t3)   # Block instr. 1: | X
       add  $v0, $t0, $t1 # Block instr. 2: | O
       jr   $ra           # Block instr. 3: | O
```

In such a case, there will be 4 instruction cache misses.

However, if we assumed that `la` consists of two instructions, we have 5 cache misses. With the given information, we accept both solutions, as long as the solution steps are clear.

- L3: Assumptions:
  - $D + W$ in the same clock cycle => D stage sees value written in W stage.
  - The erroneous part of the execution should *not* be included in this calculation.
  - Branches statically predicted not taken, misprediction solved using flushing (penalty 3 cycles).

```
start: la   $a0, numbers
       addi $a1, $0, 10
       jal  foo

foo:   addi $sp,$sp,-4   # FDEMW
       # STALL              FDEMW
       # STALL               FDEMW
       sw   $a0, 0($sp)  #    FDEMW
loop:  sw   $a1, 0($a0)
       addi $a0, $a0, 4
       addi $a1, $a1, -1
       # STALL (10 times)
       # STALL (10 times)
       bne  $a1, $0, loop
       # FLUSH 3 cycles (only when taken, 9 times)

       lw   $t3, 0($sp)
       addi $sp, $sp, 4
       # STALL
       lw   $t0, 0($t3)
       lw   $t1, 4($t3)
       # STALL
       # STALL
       add  $v0, $t0, $t1
       jr   $ra
```

Answer: Number of executed instructions is $2 + 4 * 10 + 6 = 48$. The number of clock cycles is 48 instructions $+ 5$ stalls $+ 2 * 10$ stalls $+ 3 * 9$ stalls $= 100$ clock cycles

The question is divided into three analysis tasks, each corresponding to one of three levels of difficulty: L1, L2, and L3. The question is graded in three levels S, G, and VG, with the following criteria:

- Satisfactory (S): The task at level L1 is solved correctly.
- Good (G): Either the task at level L2 or the task at level L3 is solved correctly.
- Very Good (VG): Both the tasks at level L2 and L3 are solved correctly.

If none of the three levels is achieved, the exercise is considered as failed (F).

Correction of the Exam

The examiner David Broman authored the exam questions, the correction guidelines, and the suggested solutions. The following persons took part in the correction: Saranya Natarajan, Viktor Palmkvist, Daniel Lundén, Linnea Ingmar, Gizem Çaylak, and Fredrik Lundevall.