THE UNIVERSITY OF KANSAS

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

EECS 645 – Computer Architecture

Spring 2023

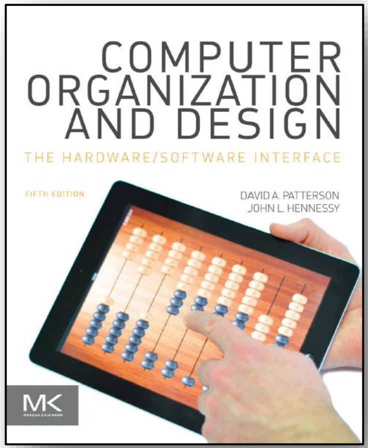
Homework 06 (MIPS ISA)

Student Name: Student ID:

Reading

- **Textbook 2**
  - **Title:** ***Computer Organization and Design: The Hardware/Software Interface***
  - **Author:** David A. Patterson, and John L. Hennessy
  - **Publisher:** Morgan Kaufmann; 5th Edition, 2014
  - **ISBN:** 978-0-12-407726-3
- **Chapters**
  - **Chapter 02:** Instructions: Language of the Computer

P&H
5th Edition


COMPUTER ORGANIZATION AND DESIGN
THE HARDWARE/SOFTWARE INTERFACE
FIFTH EDITION
DAVID A. PATTERSON
JOHN L. HENNESSY
MK

Homework Exercises

- **Homework 06**
  - **Chapter 02 of P&H 5th Edition (Textbook 2)**
  - **Problems: 2.24, 2.25, 2.26, 2.27, 2.31, 2.34, 2.46**
- **Submit to Canvas**
- **Due on Friday 03/31/2023 at 12:00PM (NOON) on Canvas**

2.24 (10 Points) [5] <§2.7> Suppose the program counter (PC) is set to `0x2000 0000`. Is it possible to use the jump (j) MIPS assembly instruction to set the PC to the address as `0x4000 0000`? Is it possible to use the branch-on-equal (beq) MIPS assembly instruction to set the PC to this same address?

2.25 (10 Points) The following instruction is not included in the MIPS instruction set:

```
rpt $t2, loop # if(R[rs]>0) R[rs]=R[rs]−1, PC=PC+4+BranchAddr
```

2.25.1 [5] <§2.7> If this instruction were to be implemented in the MIPS instruction set, what is the most appropriate instruction format?

2.25.2 [5] <§2.7> What is the shortest sequence of MIPS instructions that performs the same operation?

2.26 (10 Points) Consider the following MIPS loop:

```
LOOP: slt  $t2, $0,  $t1
      beq  $t2, $0,  DONE
      addi $t1, $t1, -1
      addi $s2, $s2, 2
      j    LOOP
DONE:
```

2.26.1 [5] <§2.7> Assume that the register `$t1` is initialized to the value 10. What is the value in register `$s2` assuming `$s2` is initially zero?

2.26.2 [5] <§2.7> For the loop above, write the equivalent C code routine. Assume that the registers `$s1`, `$s2`, `$t1`, and `$t2` are integers `A`, `B`, `i`, and `temp`, respectively.

2.26.3 [5] <§2.7> For the loop written in MIPS assembly above, assume that the register `$t1` is initialized to the value N. How many MIPS instructions are executed?

2.27 (10 Points) [5] <§2.7> Translate the following C code to MIPS assembly code. Use a minimum number of instructions. Assume that the values of `a`, `b`, `i`, and `j` are in registers `$s0`, `$s1`, `$t0`, and `$t1`, respectively. Also, assume that register `$s2` holds the base address of the array D.

```
for(i=0; i<a; i++)
   for(j=0; j<b; j++)
      D[4*j] = i + j;
```

2.31 (20 Points) [5] <§2.8> Implement the following C code in MIPS assembly. What is the total number of MIPS instructions needed to execute the function?

```
unsigned int fib(unsigned int n){
    if (n==0)
       return 0;
    else if (n == 1)
       return 1;
    else
       return fib(n-1) + fib(n-2);}
```

2.34 (20 Points) [5] <§2.8> Translate function `f` into MIPS assembly language. If you need to use registers `$t0` through `$t7`, use the lower-numbered registers first. Assume the function declaration for `func` is "`int func(int a, int b);`". The code for function `f` is as follows:

```
int f(int a, int b, int c, int d){
  return func(func(a,b),c+d);
}
```

2.46 (20 Points) Assume for a given processor the CPI of arithmetic instructions is 1, the CPI of load/store instructions is 10, and the CPI of branch instructions is 3. Assume a program has the following instruction breakdowns: 500 million arithmetic instructions, 300 million load/store instructions, 100 million branch instructions.

2.46.1 [5] <§2.19> Suppose that new, more powerful arithmetic instructions are added to the instruction set. On average, through the use of these more powerful arithmetic instructions, we can reduce the number of arithmetic instructions needed to execute a program by 25%, at the cost of increasing the clock cycle time by only 10%. Is this a good design choice? Why?

2.46.2 [5] <§2.19> Suppose that we find a way to double the performance of arithmetic instructions. What is the overall speedup of our machine? What if we find a way to improve the performance of arithmetic instructions by 10 times?