Homework 2

In your own words (five-point penalty, per problem/question, for duplicating the textbook answer)

1. Describe the general syntax of the MIPS arithmetic assembly instructions

All instructions have 3 operands. That is one destination, with two operands. The order of operands is fixed, with destination first. The code in C would be a= b + c, where in MIPS code it would be: add a, b, c.

1. Describe the MIPS load and store instructions in detail.

Instructions have different number of operands. 32 bits representing a single instruction. Load instructions read data from the memory and copy it to a register. Store instructions write data from a register to memory.

1. Describe the nature of the three MIPS machine language formats.

MIPS come in 32-bit size instructions. R-format is the arithmetic instruction format. These instructions have opcode of 0, and they have different functional values besides for the first three shift instruction in which operations registers.

The I-format is the transfer, branch, immediate format, these instructions have the opcode numbers of 2 and 3. J-format, the jump instruction format, they are opcode numbers greater than three. These are the 16 bit immediate which is sign-extended to a 32 bit besides for or, xor, and lui instructions.

1. What are the steps necessary to execute a procedure?

There are 6: Firstly, the parameters should be placed where the procedure is able to access them, then transfer the control to the procedure. After, we need to allocate the storage resources that are necessary for the required procedure. When performing the desired task, we need to put the result of the procedure value in a place, where the calling program would be able to retrieve and access it. Lastly, the control should be returned to point origin, due to the procedure being called from several locations in the program.

Register are the fastest and quickest way to hold the data make it accessible.

1. Which registers implicitly support functions of the processor?

$ra supports the implicit support functions of the processor. It holds the address of the current instructions being executed. The *jal* instruction saves program counter + 4 in register $ra which is used to link the following instructions to set up the procedure to return.

1. How do programs make use of $sp ($29)?

$sp is the stock pointer register. It consists of 1 word, so $sp is 4 byte for each push or pop operation. A stack that grows from higher to lower memory gets adjusted so PUSH: SP = SP – 4, and POP: SP = SP + 4. The stack is implemented using physical memory and located at the upper end.

1. What are the addressing modes of the MIPS assembly language?

Addressing mode is one of the listed addressing type that are specified by their multi-role use of various operands, and addresses. They are – *immediate addressing* - operand is usually a constant containing the instruction within, *register addressing -* the operand is listed as a register, *base/displacement addressing:* the operand is located at the memory location, where the address is a sum of the register and a constant, *PC-relative addressing:* where the sum of the PC and a constant is the branch address, and lastly *pseudo direct addressing* the jump address is the 26 bit of the upper bits of the PC that is concatenated with the instruction.

1. Describe “data race” or race condition.

Race condition is a error and it usually referred to as a issue that occurs in the timing of the ordering in events that lead to program behavior error. These conditions could be caused by data races, but they are rare. It is useful to locate race conditions than the data races, even though it is a difficult procedure, it makes it easier once identified.

1. Describe the steps to translate and execute a program source code written in a high-level language.

Compilers are the ones that translate code that is written in high level language into a lower level language so it can be translated to source code. It makes it executable. Complier first breaks them into assembly instructions. MIPS being able to perform one operation at a time is the reason why its broken up this way. MIPS then calculates the statement, by placing temporary variables per operations. Then the job left for the compiler is to associate program variables and the registers they belong to as well as data structures such as arrays and other structures. These translated object module in machine language are linker and combine multiple tasks to be completed. The loader locates marching code to be placed in memory location for execution.

1. Describe three principles that guide instruction set designers.

*Simplicity favors regularity* motivates many of the features in MIPS instruction set. It keeps it all to a single size, and always requires 3 registers.

*Smaller is faster*, MIPS having just 32 registers keeps this compact and fast.

*Good design demands good compromises* a perfect balance of sacrifice keeps it running faster, smoother, and available for a long time.