Homework 3

Name: Jeremy Ramilo \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Describe the 3 instruction formats (R-type, I-type, and J-type).

R- type instructions have to do with register instructions. Instructions that do not have a target address, immediate value, or branch. Three register operands

I – type instructions are immediate instructions

J – type instructions are jump type instructions. 6 bits for operation field and the rest for address field

1. How does the MIPS support only ***beq*** and ***bne***? Explain.

Beq and bne use PC- relative addressing which is a form of branch addressing. It causes the MIPS address to be relative to the address of the following instruction rather than the current instruction making the common case fast. MIPS also makes beq and bne inverses of one another.

1. Describe the effects and consequences of using the ***jal*** instruction.

Jal instruction “jumps” to an address and saves the return address in register $ra

1. Describe the three general parts of any procedure (prologue, body, and epilogue).

Prologue – puts parameters in a place where the procedure can access them, transfers control to the procedure, and acquires storage resources needed for the procedure

Body – performs the desired task and puts the value in a place where the calling program can access it

Epilogue- return control to the point of origin

1. Under what conditions is a procedure required to use the stack?

A stack is required when a procedure needs more than the four arguments and two return values.

1. Under what conditions may a procedure use the $t0-$t9 registers? Explain

When the procedure requires temporary variables. $t0-$t9 are temporary registers that are not preserved by the callee.

1. What is the nature of ***syscall***?

Syscall retrieves return values from the result registers given argument values.

1. Which is more efficient: arrays or pointers? Explain.

People can use pointers to get greater efficiency than arrays do, but modern optimizing compilers can produce code for the array version that that is just as good as using pointers. Therefore, they both equal out.

1. Will more powerful instructions result in more efficient programs? Explain.

No, not necessarily. This is because the powerful instructions may rarely be used. The powerful instructions may increase efficiency dramatically but they are not used frequently enough to make the program more efficient. They are not making the common case fast.

1. Under what circumstances is it prudent to avoid writing code in assembly language?

It is not prudent to write assembly code when we are working on large projects. Assembly would cause many challenges with maintainability and make simple functions very tedious as compared to higher level programming languages.