**Problem 2:**

2a) Li is the instruction Load Integer, which will load a specified integer into the specified register. In bare machine, this instruction becomes ori (or immediate).

La is the instruction Load Address, which will load the RAM address of the variable into the specified register. In bare machine, this instruction becomes two new instructions: lui (load unsigned integer) and ori.

2b) Main starts at memory address 0x00400024.

jal 0x00400024 [main] is where the ra is linked to the memory address directly under the instruction at 0x00400018. That way the program can look back at the return address and jump to it once finished computations.

nop #*this is where the program jumps to when factorial is complete*

ori $2, $0, 10 #*load an immediate 10 to $v0*

syscall #*print the result*

addiu $29 $29 -32 #*Shift the stack pointer down to create space for the 32-bit integers.*

sw $31 20($29) #*Store the stack pointers address in the return address*

sw $30 16($29)

addiu $30, $29, 32 #*The frame pointer points to the end of the stack*

ori $4, $0, 10 #*link an immediate value to $a0. In this case, it is 10.*

lui $1, 64 [fact] #*$t0 will have the address of where fact is called. la is link address.*

ori $8, $1, 80 [fact] #*$t0 will now have the value of where fact is.*

jalr $31, $8 #*the return address is now linked to $t0, which is 0x00400050*

lw $31, 20($29) #*load the stack pointer into the return address*

lw $30, 16($29) #*and the frame pointer*

jr $31 #*jump to the return address*

fact:

addiu $29, $29, -32 #*Shift the stack pointer down for a 32-bit integer*

sw $31, 20($29) #*save the return address on where the stack pointer was shifted.*

sw $30, 16($29) #*same with frame pointer*

addiu $30, $29, 32 #*align the frame pointer to where the integer will be stored*

sw $4, 0($30) #*store the argument value that is in $a0 where the frame pointer is.*

lw $2, 0($30) #*load $v0’s address*

addu $2, $0, $4 #*move the value of $a0 to register $v0*

bgtz $2 12 [$L2-0x0040006}#*When $v0 ==0, continue to the next instruction, else branch to $L2*

ori $2, $0, 1 #*load an immediate value 1 to $v0.*

j 0x00400090 [$L1] #*Jump to $L1*

$L2

lw $3, 0($30) #*load the address of the frame pointer for the second value. $v1 = $a0*

addiu $2, $3, -1 #*subtract $v1 by 1 and store in $v0.*

addu $4, $0, $2 #*move the new value of $v0 to $a0*

jal 0x00400050 #*jump to fact and link the next address after this instruction*

lw $3, 0($30) #*Load whatever the frame pointer pointed to into $v1*

mul $2, $2, $3 #*multiply $v0 by $v1 and store value in $v1*

$L1:

lw $31, 20($29) #*load the stack pointer into the return address*

lw $30, 16($29) #*and to the frame pointer by an offset of 16*

addiu $29, $29, 32 #*shift the stack pointer up 32 bits*

jr $31 #*Jump to the return address*