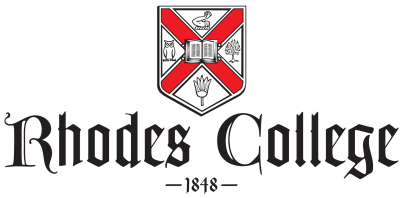
COMP 231-01
Introduction to Computer Organization
Final Exam Review



For this exam, the main addition to the exam is **functions** in assembly language and the register conventions we talked about in class. When you're designing assembly language programs on the test, you're responsible for using registers correctly according to the conventions. I recommend studying and reviewing the **register conventions** section of the assembly textbook and then working on coding the following exercises. We're also covering the FDXW cycle and memory, but those questions will appear as short answer problems.

To study for this exam, I recommend reviewing your work for the course, most especially the exam reviews and tests, as well as this final exam review. This exam is cumulative and will include questions from across the semester.

- The function detailed below is a recursive C function for calculating the result of an exponentiation operation. Note that this function is designed for integers greater than or equal to 0.

```
int exponentiate(int base, int power)
{
        if(power <= 0)
        {
                return 1;
        }
        else
        {
                return base * exponentiate(base, power-1);
        }
}
```

Translate this function into assembly language, using the calling conventions.

- The code below is an assembly language program that has been improperly translated and does not follow the calling conventions. Find and correct the errors in this program. (There are intended to be 8 errors. Some will be related to the calling conventions, but others will be related to other aspects of the program.)

```
#This function takes the values in memory location 1000 and applies the MATHFUNC to it.
#Then, this function adds that same value (from MEM 1000) with the results of the MATHFUNC operation.
_start
ldw r9, 1000

call MATHFUNC

add r0, r9, r2

MATHFUNC: #This function triples the argument value and adds 2 to it, then returns that value.
call FUNC
addi r9, 2
ret


FUNC:  #This function takes in an argument, triples the given value, and returns that value.
multi r9, r9, 3
```

Translate this function into assembly language, using the calling conventions.

- The function detailed below is a recursive C function for calculating the result of a factorial operation. Note that this function is designed for integers greater than or equal to 0.

```
int factorial(int input)
{
        if(input <= 1)
        {
                return 1;
        }
        else
        {
                return base * factorial(input - 1);
        }
}
```

Write an assembly program to calculate the factorial value of 10. To do this, write the main function and also translate this into assembly language, using the calling conventions.

- What is meant by the term spatial locality?
- What is meant by the term temporal locality?
- What is meant by the term pipelining?
- Why is out of order processing used in assembly language?
- What is the inherent conflict between branches and pipelining?

NIOS II Instruction Reference

This is a list of the syntax for the instructions that you may use from the NIOS instruction set:

instruction	usage
ldw	ldw rB, off(rA)
ldb	ldb rB, off(rA)
ldbu	ldbu rB, off(rA)
ldbu	ldbu rB, off(rA)
ldh	ldh rB, off(rA)
ldhu	ldhu rB, off(rA)
stw	stw rB, off(rA)
stb	stw rB, off(rA)
sth	stw rB, off(rA)
add	add rC, rA, rB
sub	sub rC, rA, rB
mul	mul rC, rA, rB
div	div rC, rA, rB
addi	addi rC, rA, IMMED16
subi	subi rC, rA, IMMED16
muli	muli rC, rA, IMMED16
divu	divu rC, rA, rB
and	and rC, rA, rB
or	or rC, rA, rB
xor	xor rC, rA, rB
andi	andi rC, rA, IMMED16
ori	ori rC, rA, IMMED16
xori	xori rC, rA, IMMED16
andhi	andhi rC, rA, IMMED16
orhi	orhi rC, rA, IMMED16
xorhi	xorhi rC, rA, IMMED16
mov	mov rC, rA
movi	movi rB, IMMED16
movui	movui rB, IMMED16
srl	srl rC, rA, rB
srli	srli rC, rA, IMMED5
sra	sra rC, rA, rB
srai	srai rC, rA, IMMED5
sll	sll rC, rA, rB
slli	slli rC, rA, IMMED5
jmp	jmp rA
br	br LABEL
beq	beq rA, rB, LABEL
bne	bne rA, rB, LABEL
blt	blt rA, rB, LABEL
ble	ble rA, rB, LABEL
bgt	bgt rA, rB, LABEL
bge	bge rA, rB, LABEL
bltu	bltu rA, rB, LABEL
bleu	bleu rA, rB, LABEL
bgtu	bgtu rA, rB, LABEL
bgeu	bgeu rA, rB, LABEL
call	call LABEL
ret	ret