# CO450 Computer Architectures Week 7 Exercise Handout

Recap on Signed Magnitude Notation ..... 2
Recap on Binary Excess Notation to Decimal ..... 2
Recap on Decimal to Binary Excess Notation ..... 3
Recap on Two's Complement ..... 3
Recap on Two's Complement Binary Additions ..... 4
Recap on Decimal to Excess 50 Notation ..... 4
Recap on Conversion of Decimal Numbers to SEEZMMMM Format ..... 5
Recap on Conversion of SEEZMMMM Format to Decimal Number ..... 5
Recap on Conversion of Decimal Exponent to Excess 127 Binary ..... 5
Recap on Conversion of IEEE 754 Single Precision Binary Float to Decimal ..... 5
Recap on the Little Man Computer ..... 6
The Register Transfer Language (RTL) ..... 7
MIPS Program One: Addition ..... 8
MIPS Program Two: Subtraction ..... 9
MIPS Program Three: Multiplication ..... 10
MIPS: Program Four: Division ..... 11
MIPS: Program Five: Branching Control ..... 12
MIPS: Program Six: Jumping ..... 13
The Answers ..... 14

## Recap on Signed Magnitude Notation

**1.** Represent the following decimal number in binary using Signed Magnitude Notation:

**$+107_{10}$**

| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|---|
| $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| | | | | | | | |
| | | | | | | | |

The correct answer is:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

## Recap on Binary Excess Notation to Decimal

**1**. What is the decimal number that is represented by $10111011_2$ in Excess Notation?

| | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|
| | $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| | | | | | | | | |
| Unsigned binary to decimal conversion using positional notation | | | | | | | | |
| Unsigned decimal value minus Excess ($2^{(n-1)}$) Note: *n* = number of bits | | | | | | | | |

The correct answer is:

| |
|---|
| |

## Recap on Decimal to Binary Excess Notation

**1**. What is the binary Excess Notation representation of the following decimal number:

$-63_{10}$

| Decimal plus Excess ($2^{(n-1)}$)<br>Note: *n* = number of bits | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| | $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| Convert Decimal with Excess to binary using positional notation | | | | | | | | |

The correct answer is:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

## Recap on Two's Complement

**1.** Convert $59_{10}$ to binary then use Two's Complement to convert the unsigned binary representation of $59_{10}$ in to the Two's Complemented binary representation for $-59_{10}$, what is the correct answer:

| | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|
| | $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| Positional notation used to convert decimal to binary | | | | | | | | |
| Flipped bits | | | | | | | | |
| One to add to the flipped bits above | | | | | | | | |
| Result of addition of flipped bits and one | | | | | | | | |
| *Carry Bits* | | | | | | | | |

The correct answer is:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

## Recap on Two's Complement Binary Additions

**1.** Add the following numbers together using two's complement binary representation and then answer the questions below:

**$-43_{10} + -27_{10} =$**

| | | | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| | | | | | | | | | | |
| + | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |

Did the calculation produce an overflow? YES / NO

Did the calculation produce a carryout? YES / NO

Would the calculation produce a correct result in an 8 bit system? YES / NO

How many bits were carried to the left during the calculation?

## Recap on Decimal to Excess 50 Notation

1. Represent $-16_{10}$ in excess 50 notation:

| | + | $50_{10}$ | = | |
|---|---|---|---|---|

The correct answer is:

## Recap on Conversion of Decimal Numbers to SEEZMMMM Format

1. Convert $+0.000548701_{10}$ into the SEEZMMMM format:

We have done this first one for you.

| | | |
|---|---|---|
| S = signed magnitude notation | = | |
| EE = Exponent Excess 50 notation | = | |
| ZMMMM = Mantissa | = | |
| Normalise Mantissa | = | |

The correct answer is:

| |
|---|
| |

## Recap on Conversion of SEEZMMMM Format to Decimal Number

1. Convert 04476823 in the SEEZMMMM Format to a decimal number:

We have done this first one for you.

| | | | | |
|---|---|---|---|---|
| S = signed magnitude notation | = | | = | |
| EE = Exponent Excess 50 notation | = | | = | |
| ZMMMM = Mantissa | = | | = | |

The correct answer is:

| |
|---|
| |

## Recap on Conversion of Decimal Exponent to Excess 127 Binary

1. Represent an Exponent of $-23_{10}$ in the Excess 127 format:

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Excess | 127 | = | | | | | | | | |
| Two's Complement Exponent | -23 | = | | | | | | | | |
| Result of addition of excess and exponent | | | | | | | | | | |
| *Carry's* | | | | | | | | | | |

The correct answer is:

| | | |
|---|---|---|
| | = | |

## Recap on Conversion of IEEE 754 Single Precision Binary Float to Decimal

1. Convert the following IEEE 754 single precision binary float to decimal:

$01000011110110011000000000000000_2$

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Sign | | = | | | | | | | | |
| Excess | | = | | | | | | | | |
| Convert excess binary unsigned to decimal and subtract excess 127 to get exponent | | = | | | | | | | | |

| | |
|---|---|
| Mantissa | |
| Shift decimal point with exponent | |
| Convert shifted binary mantissa to decimal | |
| Add sign to converted decimal | |

The correct answer is:

## Recap on the Little Man Computer

1. Look at the Little Man Computer code below. What number will be in the Accumulator after the CPU executes the instruction held in memory location 4?

```
            LDA   first
            STA   second
            ADD   first
            SUB   second
            OUT
stop        HLT
first       DAT   010
second      DAT
```

## The Register Transfer Language (RTL)

1. What does the following RTL notation indicate?

We have done this first one for you.

$[2_{(0:3)}] \leftarrow 1$

| |
|---|
| The RTL notation indicates that bits 0 to 3 of memory address 2 are to be set to one. |

2. What does the following RTL notation indicate?

$[4] \leftarrow [8]$

| |
|---|
| |

3. What does the following RTL notation indicate?

$[8] \leftarrow [5]+1$

| |
|---|
| |

4. What does the following RTL notation indicate?

$[PC] \leftarrow [PC]+1$

| |
|---|
| |

## MIPS Program One: Addition

The following MIPS Assembly Program was developed in the code editor MIPSter (http://www.downcastsystems.com/mipster/) and then run in the QtSpim MIPS Simulator (http://spimsimulator.sourceforge.net/)

```
# A Program to Add 10 to 10

        .data                   # Data declaration section
result: .asciiz "\n We Added 10 to 10 \n and the Result is: " #Memory location
        .text                   # Text declaration section
main:                           # Start of code section
        addi    $t0, $t0, 10    # Place immediate number 10 into t0 register
        addi    $t1, $t1, 10    # Place immediate number 10 into t1 register
        add     $t3, $t0, $t1   # Add the contents of t1 to t0 and place the result in t3
        li      $v0, 4          # Load immediate number 4 into v0 register (Print String)
        la      $a0, result     # Load memory address labelled 'result' and place into a0
        syscall                 # System call to carry out operation
        li      $v0, 1          # Load immediate number 1 into v0 register
        move    $a0, $t3        # Copy contents of register t3 to a0
        syscall                 # System call to carry out operation
        li      $v0, 10         # Load immediate number 10 into v0 register
        syscall                 # System call to carry out operation
# END OF PROGRAM
```

## MIPS Program Two: Subtraction

```
# A Program to Subtract 10 from 20
        .data                   # Data declaration section
result: .asciiz "\n We Subtracted 10 from 20 \n and the Result is: "
        .text                   # Text declaration section
main:                           # Start of code section
        addi    $t0, $t0, 20
        addi    $t1, $t1, 10
        sub     $t3, $t0, $t1   # Subtract contents of t1 from t0 and place result in t3
        li      $v0, 4
        la      $a0, result
        syscall
        li      $v0, 1
        move    $a0, $t3
        syscall
        li      $v0, 10
        syscall
# END OF PROGRAM
```

## MIPS Program Three: Multiplication

```
# A Program to Multiply 10 by 10
        .data                   # Data declaration section
result: .asciiz "\n We Multiply 10 by 10 \n and the Result is: "
        .text                   # Text declaration section
main:                           # Start of code section
        addi $t0, $t0, 10
        mult $t0, $t0           # Multiply t0 by t0
        mfhi $t1                # Move contents of Hi register to t1
        mflo $t2                # Move contents of Lo register to t2
        li $v0, 4
        la $a0, result
        syscall
        li $v0, 1
        move $a0, $t2
        syscall
        li $v0, 10
        syscall
# END OF PROGRAM
```

## MIPS: Program Four: Division

```
# A Program to Divide 10 by 3
        .data                   # Data declaration section
result:  .asciiz "\n We Divide 10 by 3 \n and the Result is: "
remain: .asciiz "\n Remainder: "
        .text                   # Text declaration section
main:                           # Start of code section
        addi $t0, $t0, 10
        addi $t1, $t1, 3
        div $t0, $t1            # Divide the contents of t0 by t1
        mfhi $t2
        mflo $t3
        li $v0, 4
        la $a0, result
        syscall
        li $v0, 1
        move $a0, $t3
        syscall
        li $v0, 4
        la $a0, remain
        syscall
        li $v0, 1
        move $a0, $t2
        syscall
        li $v0, 10
        syscall
# END OF PROGRAM
```

## MIPS: Program Five: Branching Control

```
# A Program to Count Up to 10
        .data                   # Data declaration section
count:  .asciiz  "\n Look, I can count to 10: "
end:    .asciiz  "\n Well done me!"
new:    .asciiz  "\n"
        .text                   # Text declaration section
main:                           # Start of code section
        addi    $t0, $t0, 10
        addi    $t1, $t1, 0
        li      $v0, 4
        la      $a0, count
        syscall
        li      $v0, 4
        la      $a0, new
        syscall
loop:   addi    $t0, $t0, -1
        addi    $t1, $t1, 1
        li      $v0, 1
        move    $a0, $t1
        syscall
        li      $v0, 4
        la      $a0, new
        syscall
        blez    $t0, out        # Branch if t0 contents less than or equal to zero
        b       loop            # Branch always
out:    li      $v0, 4
        la      $a0, end
        syscall
        li      $v0, 10
        syscall
# END OF PROGRAM
```

## MIPS: Program Six: Jumping

```
# A Program that Calls a Function
        .data                   # Data declaration section
enter:  .asciiz "\n Please Enter a Number: "
result: .asciiz "\n The result of adding the two numbers together is: "
        .text                   # Text declaration section
main:                           # Start of code section
        jal userent             # Jump to address labelled ‘userent’
        move $t1, $t0
        jal userent
        move $t2, $t0
        add $t3, $t1, $t2
        li $v0, 4
        la $a0, result
        syscall
        li $v0, 1
        move $a0, $t3
        syscall
        li $v0, 10
        syscall
userent:
        li $v0, 4
        la $a0, enter
        syscall
        li $v0, 5
        syscall
        move $t0, $v0
        jr $ra                  # Jump to contents of return address register
# END OF PROGRAM
```

## The Answers

Signed Magnitude Notation

1. $01101011_2$

Binary Excess Notation to Decimal

1. $+59_{10}$

Decimal to Binary Excess Notation

1. $01000001_2$

Two's Complement

1. $11000101_2$

Two's Complement Binary Additions

1. No, Yes, Yes, 4

Decimal to Excess 50 Notation

1. $+34_{10}$

Conversion of Decimal Numbers to SEEZMMMM Format

1. 04754870

Conversion of SEEZMMMM Format to Decimal Number

1. $+0.00000076823_{10}$

Conversion of Decimal Exponent to Excess 127 Binary

1. $01101000_2 = 104_{10}$

Conversion of IEEE 754 Single Precision Binary Float to Decimal

1. $+435_{10}$

Recap on the Little Man Computer

1. 10

The Register Transfer Language

1. The RTL notation indicates that bits 0 to 3 of memory address 2 are to be set to one.

2. The RTL notation indicates that the contents of memory address 8 are transferred (copied) to memory address 4

3. The RTL notation indicates that the contents of memory address 5 are incremented by 1 and the result placed in to memory address 8

4. The RTL notation indicates that the Program Counter (PC) register will be incremented by 1.