CIS 600

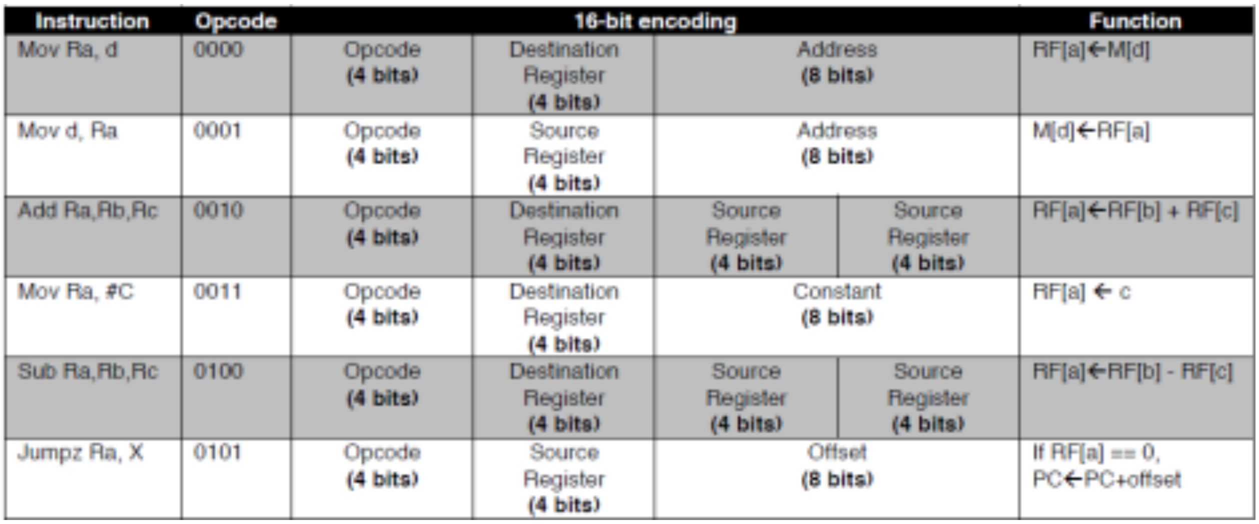
Advanced Computer Architecture

HW# 1

Due: 2/16 (Friday)
Turn in the answer to Blackboard.

Name:

Question 1 (30pts) The instruction encodings for the 6-instruction processor are as shown in the table below:



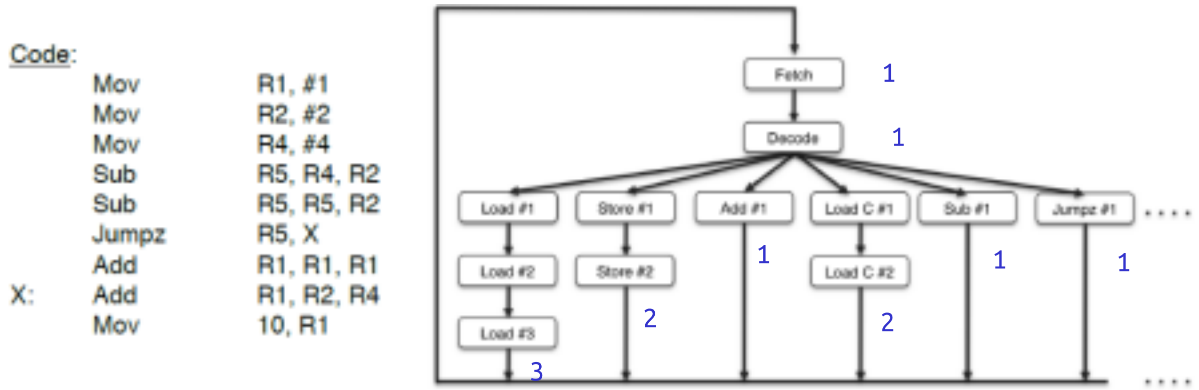
Instruction	Opcode	16-bit encoding				Function
Mov Ra, d	0000	Opcode **(4 bits)**	Destination Register **(4 bits)**	Address **(8 bits)**		RF[a]←M[d]
Mov d, Ra	0001	Opcode **(4 bits)**	Source Register **(4 bits)**	Address **(8 bits)**		M[d]←RF[a]
Add Ra,Rb,Rc	0010	Opcode **(4 bits)**	Destination Register **(4 bits)**	Source Register **(4 bits)**	Source Register **(4 bits)**	RF[a]←RF[b] + RF[c]
Mov Ra, #C	0011	Opcode **(4 bits)**	Destination Register **(4 bits)**	Constant **(8 bits)**		RF[a] ← c
Sub Ra,Rb,Rc	0100	Opcode **(4 bits)**	Destination Register **(4 bits)**	Source Register **(4 bits)**	Source Register **(4 bits)**	RF[a]←RF[b] - RF[c]
Jumpz Ra, X	0101	Opcode **(4 bits)**	Source Register **(4 bits)**	Offset **(8 bits)**		If RF[a] == 0, PC←PC+offset

a) Assume that you want to augment this ISA to support 20 additional and unique instructions (e.g. Mult, And, Or, etc.), while still keeping the instruction encoding as 16 bits. How will the execution and encoding of the Add instruction be affected? (Other instructions could be affected too, but you just need to comment on how the Add instruction will be impacted.)

Since there will be 20 additional and unique instructions, the opcode will range from 0 to 25, we need at least 5 bits to encode all 26 opcode, instead of 4 bits.
Since we need to keep instruction encoding as 16 bits, the bits for other parts encoding need to be one bit less.

For Add instruction, only 0-7 register allowed for either Destination register, or Source register

b) Now, assume that the 20 new instructions have all been added. Their addition has resulted in changes to the state diagram. A portion of the new finite state diagram is shown below. Given this new state diagram, how many clock cycles will the code (that is also shown below) take to run?



```
Code:
        Mov       R1, #1
        Mov       R2, #2
        Mov       R4, #4
        Sub       R5, R4, R2
        Sub       R5, R5, R2
        Jumpz     R5, X
        Add       R1, R1, R1
X:      Add       R1, R2, R4
        Mov       10, R1
```

```
   Mov     R1, #1          4       <---load 1 into R1 takes 4 CCs
   Mov     R2, #2          4       <---load 2 into R2 takes 4 CCs
   Mov     R4, #4          4       <---load 4 into R4 takes 4 CCs
   Sub     R5, R4, R2      3       <---R5=R4-R2=2, Sub takes 3 CCs
   Sub     R5, R5, R2      3       <---R5=R5-R2=0, Sub takes 3 CCs
   Jumpz   R5,X            3       <---R5=0, jump on zero, to label X, jumpz takes 3 CCs
   Add     R1, R1, R1              <---skip, no fetch, no decode
X: Add     R1, R2, R4      3       <---R1=R2+R4=6, Add takes 3 CCs
   Mov     10, R1          4       <---M(10)=R1, Store takes 4 CCs             Total = 28 CCs
```

Question 2 (20pts), Review:True or False (fill in T or F)

1. (___F___) Register is implemented inside the main memory on computer board

2. (___T___) von Neumann architecture has data and instructions in the same memory space.

3. (___T___) Static RAM is typically used to implement Cache

4. (___F___) Dynamic RAM is faster than Static RAM in access time.

5. (___T___) In modern computers, CPU and Memory are connected by BUS

6. (___F___) Direct Mapping in Cache system is more complex in implementation than Set Associative Mapping

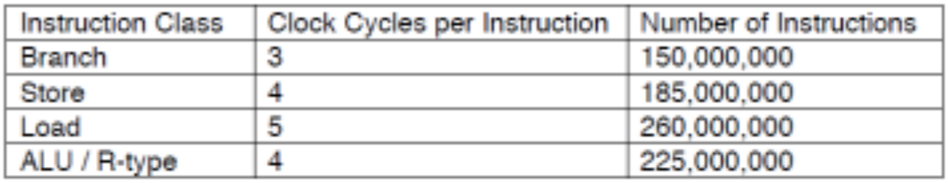
7. (___F___) Floating point data points are uniformly represented on the real line.

8. (___T___) unsigned integer data points are uniformly represented on the real line.

9. (___T___) Read and write operations can be performed on EPROM devices

10. (___F___) The density of data on the hard disk does not affect data access speed.

Question 3 (20pts) Assume that to spell check a large file, 820,000,000 instructions are needed. The instructions in the program are broken down into 4 different classes, and each class requires N clock cycles to execute. Specific information is given in the table below.



Instruction Class	Clock Cycles per Instruction	Number of Instructions
Branch	3	150,000,000
Store	4	185,000,000
Load	5	260,000,000
ALU / R-type	4	225,000,000

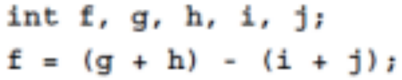
If the total execution time for this program is found to be 1.57 seconds, what is the clock cycle time of the computer on which it was run?

```
total CCs = 3*1.5*10^8 + 4*1.85*10^8 + 5*2.6*10^8 + 4*2.25*10^8
          = 3.39*10^9

time for each CC = 1.57 seconds / 3.39*10^9
                 = 0.463 * 10^(-9) seconds
                 = 0.463 nano-seconds      The machine is a 2.16 GHz machine
```

Question 4 (30pts) Write the MIPS assembly for the following C-code:

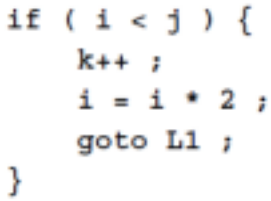
a)



```
int f, g, h, i, j;
f = (g + h) - (i + j);
```

```
suppose f, g, h, i, j are saved             MIPS Code
in memory address starting from             -------------------
1000 consecutively, that is                 lw  $1, 1004($0)
M(1000) = f                                 lw  $2, 1008($0)
M(1004) = g                                 add $1, $1, $2
M(1008) = h                                 lw  $2, 1012($0)
M(1012) = i                                 lw  $3, 1016($0)
M(1016) = j                                 add $2, $2, $3
                                            sub $1, $1, $3
the code will be:                           sw  $1, 1000($0)
```

b)



```
if ( i < j ) {
     k++ ;
     i = i * 2 ;
     goto L1 ;
}
```

```
MIPS Code
-----------------------
        sub  $2, $1, $2 <---- $2=j-i
        blez $2, X      <---- branch to X if j-i<=0
        addi $3, $3, 1  <---- $3=$3+1
        sll  $1, $1, 1  <---- $1=i*2
        jr   $31        <---- jump to L1
L1:     ....
        ....
X :     ....            <---- if i>=j, will branch here
        ....
```

```
suppose i, j, k are loaded
in register $1, $2, $3,
L1 is loaded in $31

the code will be:
```

c)

i = 0;
While (i != 5)
i = i + 5;

```
MIPS Code
-----------------------
        and  $1, $1, $0         <----$1=0
        addi $2, $0, 5          <----$2=5
Loop:                           <----jump here until $1=5
        beq  $1, $2, End        <----if $1=$2=5, goto End
        addi $1, $1, 5          <----$1=$1+5
        j    L1                 <----jump to L1
End:                            <----end of loop
```