PART 1

(1) MIPS ENCODING

(1.1) srl $t3, $t4, 10

INSTRUCTION CODING FORMAT: R-TYPE

OPCODE (6 BITS)	rs (5 bits)	rt (5 bits)	rd (5 bits)	SHAMT (5 bits)	FUNCT (6 bits)

srl rd, rt, shamt

OPCODE: 000000 (OPCODE FOR R-TYPE INSTRUCTIONS IS ALWAYS ZERO)

rs: 00000 (UNUSED FIELDS CODED WITH ALL 0 BITS)

rt: $t4 = 12

$t0 = 8; $t1 = 9; $t2 = 10; $t3 = 11; $t4 = 12

rd: $t3 = 11

SHAMT: 10

FUNCT: 2 (FROM MIPS REFERENCE DATA CARD)

DECIMAL TO BINARY CONVERTIONS:

rt:

12 | 2
0 ← 6 | 2
0 ← 3 | 2
1 ← 1

$12_{10} = 1100_2$

rd:

11 | 2
1 ← 5 | 2
1 ← 2 | 2
0 ← 1

$11_{10} = 1011_2$

SHAMT:

10 | 2
0 ← 5 | 2
1 ← 2 | 2
0 ← 1

$10_{10} = 1010_2$

FUNCT:

2 | 2
0 ← 1

$2_{10} = 10_2$

BIN TO HEX

0000 = 0x0
0001 = 0x1
0010 = 0x2
0011 = 0x3
0100 = 0x4
0101 = 0x5
0110 = 0x6
0111 = 0x7
1000 = 0x8
1001 = 0x9
1010 = 0xA
1011 = 0xB
1100 = 0xC
1101 = 0xD
1110 = 0xE
1111 = 0xF

REPLACING BINARY NUMBERS ON HIS FIELD:

OPCODE	rs	rt	rd	SHAMT	FUNCT
000000	00000	01100	01011	01010	000010
0x0	0x0	0x0	0xC	0x5	0xA	0x8	0x2

FIELDS WITH DECIMAL VALUES

OPCODE	rs	rt	rd	SHAMT	FUNCT
0	0	12	11	10	2

MACHINE CODE: 0x000C5A82

1.2 lw $t0, 32($s1)

INSTRUCTION CODING FORMAT : I-TYPE

OPCODE (6 bits)	rs (5 bits)	rt (5 bits)	IMMEDIATE (16 bits)

lw rt, IMMEDIATE(rs)

OPCODE : 0x23 (FROM MIPS REFERENCE DATA CARD) 00|10 0011

rs : $s1 = 17 $s0 = 16 ; $s1 = 17

rt : $t0 = 8

IMMEDIATE : 32

DECIMAL TO BINARY CONVERTIONS:

rs:

17 |2
1 8 |2
0 4 |2
0 2 |2
0 ← 1

$17_{10} = 10001_2$

rt:

8 |2
0 4 |2
0 2 |2
0 ← 1

$8_{10} = 1000_2$

IMMEDIATE: 32

32 |2
0 16 |2
0 8 |2
0 4 |2
0 2 |2
0 ← 1

$32_{10} = 100000_2$

REPLACING BINARY NUMBERS ON HIS FIELD:

OPCODE	rs	rt	IMMEDIATE
100011	10001	01000	0000000000100000
0x8	0xE	0x2	0x8 0x0 0x0 0x2 0x0

OPCODE BINARY TO DECIMAL CONVERTION:

$100011_2 = 1\times2^5 + 0\times2^4 + 0\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 = 32 + 0 + 0 + 0 + 2 + 1$

$= 35$

FIELDS WITH DECIMAL VALUES:

OPCODE	rs	rt	IMMEDIATE
35	17	8	32

MACHINE CODE : 0x8E280020

(2) MIPS DECODING

(2.1) $0x AE09 0020

* WHAT IS THE OPCODE (IN ASSEMBLY) OF THIS INSTRUCTION?

$0xA = 1010_2$

$0xE = 1110_2$

$0xAE = \underbrace{101011}_{\text{OPCODE}}10_2$

$\underbrace{101011}_{0xB}{}_2 = 0x2B$

ACCORDING TO THE MIPS REFERENCE DATA CARD THE OPCODE IN ASSEMBLY FOR $0x2B IS SW

* WHAT TYPE (I/J/R) IS THIS INSTRUCTION?

ACCORDING TO THE MIPS REFERENCE DATA CARD THE INSTRUCTION TYPE IS I-TYPE

* WHICH REGISTERS ARE READ/UPDATED IN EXECUTING THIS INSTRUCTION? YOU CAN USE EITHER ASSEMBLY NAMES OR REGISTER NUMBERS IN YOUR ANSWER

$0xAE09 0020 = 10101 10000 01001 0000000000100000_2$

I-TYPE INSTRUCTION FIELDS:

OPCODE (6 bits)	rs (5 bits)	rt (5 bits)	IMMEDIATE (16 bits)
101011	10000	01001	0000000000100000

BINARY TO DECIMAL CONVERTIONS:

rs: $10000_2 = 1\times2^4 + 0\times2^3 + 0\times2^2 + 0\times2^1 + 0\times2^0 = 16 + 0 + 0 + 0 + 0$
$= 16_{10}$

rt: $01001_2 = 0\times2^4 + 1\times2^3 + 0\times2^2 + 0\times2^1 + 1\times2^0 = 0 + 8 + 0 + 0 + 1$
$= 9_{10}$

REGISTERS USED: $9 OR $t1, AND $16 OR $s0

* WHAT IS THE ADDRESS OF THE NEXT INSTRUCTION WE WILL FETCH AND EXECUTION? IF THE INSTRUCTION IS A CONDITIONAL BRANCH, GIVE TWO POSSIBLE ADDRESSES. GIVE THE ANSWER AS A HEXADECIMAL VALUE AND SHOW THE ADDRESS CALCULATION STEPS TO GET FULL CREDIT.

ASSUMING THE MIPS INSTRUCTION IS AT ADDRESS
0x 00800100

NEXT INSTRUCTION ADDRESS:
0x 00800100 + 0x 4 = 0x 00800104

(2.2) 0x12530011

* WHAT IS THE OPCODE (IN ASSEMBLY) OF THIS INSTRUCTION?

$0x1 = 0001_2$

$0x2 = 0010_2$

$0x12 = \underbrace{0001\,00}_{\text{OPCODE}}10_2$

$\underbrace{000100}_{0x4}{}_2 = 0x4$

ACCORDING TO THE MIPS REFERENCE DATA CARD THE OPCODE IN ASSEMBLY FOR 0x4 IS beq

* WHAT TYPE (I/J/R) IS THIS INSTRUCTION?

ACCORDING TO THE MIPS REFERENCE DATA CARD THE INSTRUCTION TYPE IS I-TYPE

* WHICH REGISTERS ARE READ/UPDATED IN EXECUTING THIS INSTRUCTION? YOU CAN USE EITHER ASSEMBLY NAMES OR REGISTER NUMBERS IN YOUR ANSWER

0x12530011 = 00010010 10 10011 0000000000010001

I-TYPE INSTRUCTION FIELDS:

OPCODE (6 bits)	rs (5 bits)	rt (5 bits)	IMMEDIATE (16 bits)
000100	10010	10011	0000000000010001

BINARY TO DECIMAL CONVERTIONS:

$rs: 10010_2 = 1\times2^4 + 0\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0 = 16+0+0+2+0$

$= 18_{10}$

$rt: 10011_2 = 1\times2^4 + 0\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 = 16+0+0+2+1$

$= 19_{10}$

$s0 = 16$; $s1 = 17$; $s2 = 18$; $s3 = 19$

REGISTERS USED: $18 OR $s2, AND $19 OR $s3

* WHAT IS THE ADDRESS OF THE NEXT INSTRUCTION WE WILL FETCH AND EXECUTION? IF THE INSTRUCTION IS A CONDITIONAL BRANCH, GIVE TWO POSSIBLE ADDRESSES. GIVE THE ANSWER AS A HEXADECIMAL VALUE AND SHOW THE ADDRESS CALCULATION STEPS TO GET FULL CREDIT.

$PC = 0x00800100$

ADDRESS AFTER FALSE CASE:

$NewAddr = PC + 4 = 0x00800100 + 0x4 = 0x00800104$

ADDRESS AFTER TRUE CASE:

IMMEDIATE: $0000\,0000\,0000\,0001_2 = 0x0011$

$NewAddr = (PC + 4) + 4 * IMMEDIATE$

$= (0x00800100 + 0x04) + 0x04 * 0x11$

$= 0x00800104 + 0x44$

$$\begin{array}{r} 0x00800104 \\ +\quad 0x44 \\ \hline 0x00800148 \end{array} \longrightarrow \text{New Addr}$$

THE TWO POSSIBLE ADDRESSES ARE:

$0x00800104$ AND $0x00800148$

(2.3) $0x08000AABB$

* WHAT IS THE OPCODE (IN ASSEMBLY) OF THIS INSTRUCTION?

$0x0 = 0000_2$

$0x8 = 1000_2$

$0x08 = \underbrace{000010}_{\text{OPCODE}}000_2$

$000010_2 = 0x2$

ACCORDING TO THE MIPS REFERENCE DATA CARD THE OPCODE IN ASSEMBLY FOR $0x2$ IS j

* WHAT TYPE (I/J/R) IS THIS INSTRUCTION?

ACCORDING TO THE MIPS REFERENCE DATA CARD THE INSTRUCTION TYPE IS J-TYPE

* WHICH REGISTERS ARE READ/UPDATED IN EXECUTING THIS INSTRUCTION?

THIS INSTRUCTION DO NOT READ OR UPDATE ANY REGISTER.

* WHAT IS THE ADDRESS OF THE NEXT INSTRUCTION WE WILL FETCH AND EXECUTION?

J INSTRUCTION FORMAT:

OPCODE (6 bits)	ADDRESS (26 bits)
000010	00000000000101010101011011

$0x08000AABB = 00001000000000000101010101011011$

OPERATION OF JUMP INSTRUCTION: PC = JumpAddr

JumpAddr = { PC+4[31:28], ADDRESS, 2'b0 }

$PC+4 = 0x00800100 + 0x04 = 0x00800104$

$PC+4[31:28] = 0000_2$

ADDRESS = 0000000000101010101011011

2'b0 = 00

Jump Addr = {0000,0000,0000,0010,1010,1010,1110,1100}
0x0 0x0 0x0 0x2 0xA 0xA 0xE 0xC

THE ADDRESS OF THE NEXT INSTRUCTION IS:
0x0002AAEC

(3) CONSIDER THE FOLLOWING MIPS CODE

```
LOOP: slt $t2, $0, $t1
      beq $t2, $0, DONE
      addi $t1, $t1, -2
      addi $s2, $s2, 1
      j LOOP
DONE:
```

ASSUMING THAT $t1 = 20 AND $s2 = 0 INITIALLY

(1) WHAT IS THE FINAL VALUE OF $s2?

THE LOOP FINISH WHEN $t1 == 0, THAT HAPPENS WHEN $t1 = 20 - 2·(10), (FROM addi $t1, $t1, -2) ← THIS LINE MUST BE REPEATED 10 TIMES TO ACHIEVE $t1 == 0, ALSO THE LINE addi $s2, $s2, 1 REPEATS 10 TIMES, SO THE FINAL VALUE OF $s2 IS TEN (10)

(2) HOW MANY MIPS INSTRUCTIONS WILL BE EXECUTED FOR THE GIVEN SEQUENCE? FROM THE FIRST QUESTION, THE 5 INSTRUCTIONS INSIDE THE LOOP ARE REPEATED 10 TIMES EACH ONE, PLUS ONE MORE TIME FOR THE TWO FIRST INSTRUCTIONS (50+2) WHEN $t1==0 THEN $t2 BECOMES ZERO, THE CONDITION IS TRUE AND THE JUMP TO DONE LABEL IS EXECUTED FINISHING THE SEQUENCE. THE ANSWER IS 52

(4) TRANSLATE FUNCTION FOO INTO MIPS ASSEMBLY LANGUAGE. YOU MUST FOLLOW MIPS REGISTER USAGE CONVENTIONS FOR FUNCTION CALLS/RETURNS AS DISCUSSED IN CLASS. ASSUME THE FUNCTION DECLARATION OF BAR IS "INT BAR(INT X);" AND A LABEL BAR MARKS THE STARTING POINT OF ITS FUNCTION BODY. YOU CAN ASSUME THAT FUNCTION BAR FOLLOWS THE SAME MIPS CONVENTIONS BUT DO NOT NEED TO IMPLEMENT IT.

```
void FOO(INT *VALS){
    WHILE(BAR(*VALS) != 0){
        VALS++;
    }
}
```

MIPS:

```
FOO:    ADDI  $SP, $SP, -32   # MAKING SPACE ON THE STACK
        SW    $RA, 28($SP)    # STORING $RA ON THE STACK
        SW    $S0, 24($SP)    # STORING $S0 ON THE STACK
        ADDU  $S0, $0, $A0    # COPYING $A0 (FOO ARGUMENT) TO $S0
WHILE:  LW    $A0, ($S0)      # LOADING THE CONTENT OF $S0 ADDRESS ON $A0
        JAL   BAR             # CALL TO BAR FUNCTION WITH $A0=*VALS AS ARGUMENT
        BEQ   $V0, $0, FOOEND # WHILE (BAR(*VALS) != 0)
        ADDI  $S0, $S0, 4     # VALS++; INCREMENT ADDRESS TO NEXT INT
        J     WHILE           # JUMP TO WHILE LABEL TO REPEAT
FOOEND: LW    $S0, 24($SP)    # READ FROM STACK AND STORE TO $S0
        LW    $RA, 28($SP)    # READ FROM STACK AND STORE TO $RA
        ADDI  $SP, $SP, 32    # RESTORE $SP VALUE BEFORE FOO CALL
        JR    $RA             # JUMP TO INSTRUCTION NEXT TO FOO CALL
```