Module 3 Assignment: MIPS Assembly Language

EN.605.204 Computer Organization

Question #1: Assembling MIPS Instructions

Assemble (MIPS -> binary/hex) the following instructions into their binary and hex equivalents. Please show your work.

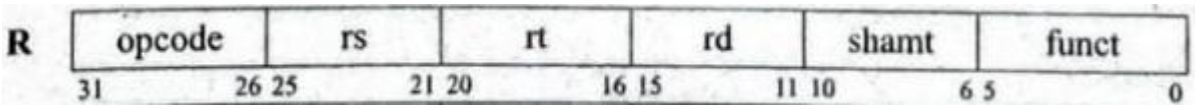
1. and $t7, $t6, $t0
2. or $s3, $s2, $s0
3. andi $s0, $t4 0x9FB6
4. jr $ra

1. Jump j J PC=JumpAddr (5) 2_{hex}

look up and instructions:

And and R R[rd] = R[rs] & R[rt] $0 / 24_{hex}$

Then, look up basic instruction formats:



OP code= 0 in decimal = 000000 in binary

Then, decode the registers:

rs= $t6=14 in decimal, 14=2^3 + 2^2+2^1= 01110 in binary

rt=$t0=8 in decimal, 8=2^3 = 01000 in binary

rd=$t7= 15 in decimal, 15= 2^3+2^2+2^1+2^0=01111 in binary

shamt=0 in decimal = 00000 in 5 bit binary

funct=24 in hex = 2*16^1+4*16^0= 36 in decimal = 2^5+2^2=100100 in 6 bit binary.

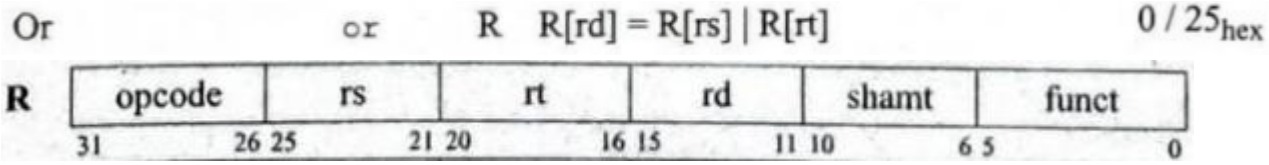
Binary equivalent: 00000001110010000111100000100100

000000	01110	01000	01111	00000	100100
opcode	rs	rt	rd	shamt	func

Hex equivalent: 0x01C87824

0000	0001	1100	1000	0111	1000	0010	0100
0	1	C	8	7	8	2	4

2. look up instructions:



OP code= 0 in decimal = 000000 in binary

Then, decode the registers:

rs= $s2=18 in decimal, 18=2^4 + 2^1= 10010 in binary

rt=$s0=16 in decimal, 16=2^4 = 10000 in binary

rd=$s3= 19 in decimal, 19= 2^4+2^1+2^0=10011 in binary

shamt=0 in decimal = 00000 in 5 bit binary

funct=25 in hex = 2*16^1+5*16^0= 37 in decimal = 2^5+2^2+2^0=100101 in 6 bit binary.

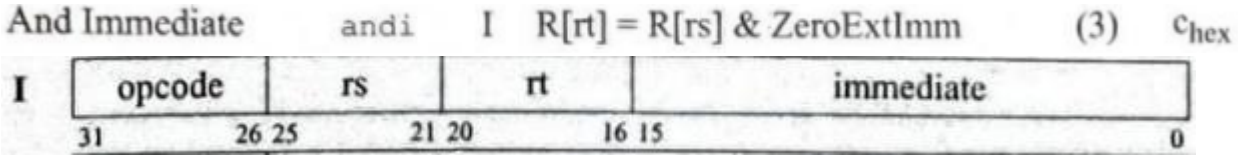
Binary equivalent: 00000010010100001001100000100101

000000	10010	10000	10011	00000	100101
opcode	rs	rt	rd	shamt	func

Hex equivalent: 0x02509825

0000	0010	0101	0000	1001	1000	0010	0101
0	2	5	0	9	8	2	5

3. Look up instructions:



OP code= C in hex= 12 in decimal = 001100 in binary

Then, decode the registers:

rs= $t4=12 in decimal, 12=2^3 + 2^2= 01100 in binary

rt=$s0=16 in decimal, 16=2^4 = 10000 in binary

immediate=0x9FB6=9*16^3+15*16^2+11*16^1+6*16^0=2^15+2^12+2^11+2^10+2^9+2^8+2^7+2^5+2^4+2^2+2^1= 1001111110110110

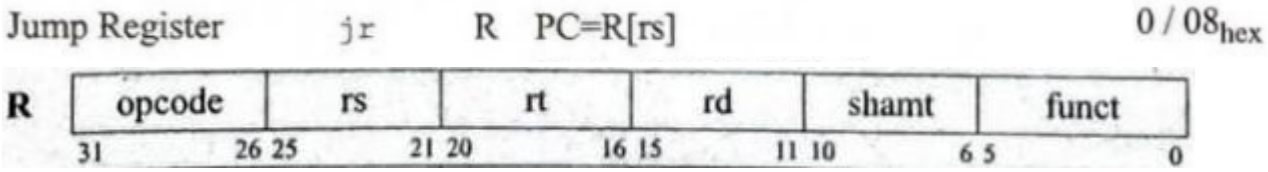
Binary equivalent: 00110001100100001001111110110110

001100	01100	10000	1001111110110110
opcode	rs	rt	immediate

Hex equivalent: 0x31909FB6

0011	0001	1001	0000	1001	1111	1011	0110
3	1	9	0	9	F	B	6

4. Look up instructions:



OP code= 0 in decimal = 000000 in binary

Then, decode the registers:

rs= $ra=31 in decimal =2^4 + 2^3 + 2^2+2^1+2^0= 11111 in binary

rt=0 = 00000 in binary

rd=0=00000 in binary

shamt=0=00000 in binary

funct=0x08=8*16^0=8 in decimal= 2^3=001000 in binary

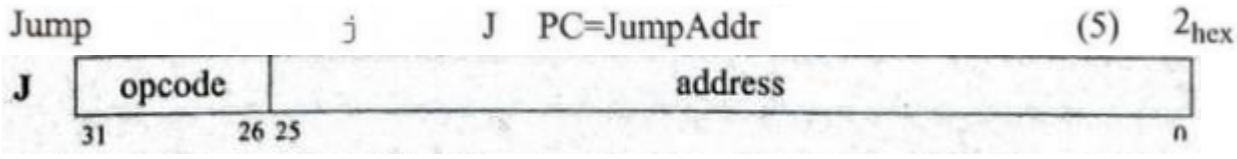
Binary equivalent: 00000011111000000000000000001000

000000	11111	00000	00000	00000	001000
opcode	Rs	rt	rd	shamt	Func

Hex equivalent: 0x03E00008

0000	0011	1110	0000	0000	0000	0000	1000
0	3	E	0	0	0	0	8

5. Loop up instructions:



OP code=2 in hex=2^1=000010

Address= 0x4567=4*16^3+5*16^2+6*16^1+7*16^0=2^14+2^10+2^8+2^6+2^5+2^2+2^1+2^0=
00000000000100010101100111

Binary equivalent: 00001000000000000100010101100111

000010	00000000000100010101100111
opcode	address

Hex equivalent: 0x08004567

0000	1000	0000	0000	0100	0101	0110	0111
0	8	0	0	4	5	6	7

Question #2: Disassembling MIPS Instructions

Disassemble (binary/hex -> MIPS) the following instructions from binary/hex into their MIPS equivalents. Please show your work.

1. 00000010100101101000100000100010
   Hex equivalent: 0x02968822
   First determine the opcode: 000000 so it is a R instruction
   Then, look at the last 6 digit to determine the funct
   100010=34=22 in hex = sub

Subtract	sub	R	R[rd] = R[rs] - R[rt]	(1) $0 / 22_{hex}$

   Determine rs=10100=20=$s4
   Determine rt=10110=22=$s6
   Determine rd=10001=17=$s1
   Determine shamt=00000=0
   Conclusion: SUB $s1 $s4 $s6
2. 0x01684824
   Binary equivalent: 00000001011010000100100000100100
   First determine the opcode: 000000 so it is a R instruction
   Then, look at the last 6 digit to determine the funct
   100100= 20 in hex = add

Add	add	R	R[rd] = R[rs] + R[rt]	(1) $0 / 20_{hex}$

Then, determine the registers:

Determine rs= 01011=11=$t3

Determine rt= 01000=8=$t0

Determine rd=01001=9=$t1

Determine shamt=00000=0

Conclusion: AND $t1 $t3 $t0

3. 00110110110011000100001100100001

   Hex equivalent: 0x36CC4321

   First look at the opcode: 001101=13=d in hex= OR Immediate

Or Immediate	ori	I	R[rt] = R[rs] | ZeroExtImm	(3)	d_{hex}

   Then, determine the registers:

   Determine rs=10110=22=$s6

   Determine rt=01100=12=$t4

   Determine immediate=0100/0011/0010/0001=0x4321

   Conclusion: ORI $t4 $s6 0x4321

4. 0x2288BEEF

   Binary equivalent: 00100001010001000101111101101111

   First, look at the opcode:001000=8 in hex=ADDI

Add Immediate	addi	I	R[rt] = R[rs] + SignExtImm	(1,2)	8_{hex}

   Then, determine the registers:

   Determine rs=10100=20=$s4

   Determine rt=01000=8=$t0

   Determine immediate=1011/1110/1110/1111=0xBEEF

   Conclusion: ADDI $t0 $s4 0xBEEF

Module 3 Assignment: MIPS Assembly Language

EN.605.204 Computer Organization

Question #3: Endianness

Assemble the following instructions from MIPS to hex and give the hex representation in both big- and little-endian format:

1. **or $t7, $t2, $s4**

   opcode= 000000

   rs=$t2=01010

   rt=$s4=10100

   rd=$t7=01111

   shamt=00000

   funct=100101

   Binary equivalent= 0000/0001/0101/0100/0111/1000/0010/0101

   Hex equivalent= 0x01547825

   Order of significance=01>54>78>25

2. **andi $t6, $s1, 0x7050**

   opcode= 001100

   rs=$s1=10001

   rt=$t6=01110

   immediate=0x7050= 0111/0000/0101/0000

   Binary equivalent= 0011/0010/0010/1110/0111/0000/0101/0000

   Hex equivalent= 0x322E7050

   Order of significance=32>2E>70>50

or (little)	25	78	54	01
Memory Address	0x0	0x1	0x2	0x3

or (big)	01	54	78	25
Memory Address	0x0	0x1	0x2	0x3

andi (little)	32	2E	70	50
Memory Address	0x3	0x2	0x1	0x0

andi (big)	50	70	2E	32
Memory Address	0x3	0x2	0x1	0x0

Module 3 Assignment: MIPS Assembly Language

EN.605.204 Computer Organization

Question #4: Registers

Please answer each of the following questions and justify your answer when needed.

1. To which value is the $zero register hard-coded?
2. Which register would you use to return a value from a function call?
3. Which two registers are used for storing data on the call stack?
4. What is the difference between 's' and 't' registers?
5. Should the $at register be used by developers? Why/why not?

1. According to MIPS Green Sheet, $zero hard-coded the constant number zero
2. According to Module3 part2-slides and the MIPS Green Sheet, we use $v0 and $v1 for function call return values hence return a value from a function call
3. According to Module3 part2-slides and the MIPS Green Sheet, $s0-$s8 and $t0-$t9 are used to storing data on the call stack
4. According to Module3 part2-slides and the MIPS Green Sheet, 's' register stands for saved temporary value which is guaranteed to be the same after return, 't' register stands for temporary value which is not guaranteed to be the same after return.
5. According to Module3 part2-slides and video, $at is used for temporary values within pseudo commands. It is not preserved across function calls. According to this overstack post: https://electronics.stackexchange.com/questions/136916/use-of-at-register-in-mips You can still use $at in rare case. But the value won't be the same across function calls

Module 3 Assignment: MIPS Assembly Language

EN.605.204 Computer Organization

Deliverables

Please submit your assignment showing all work in a PDF named <Your JHEDID>_module3.pdf using the Assignment link in Canvas.