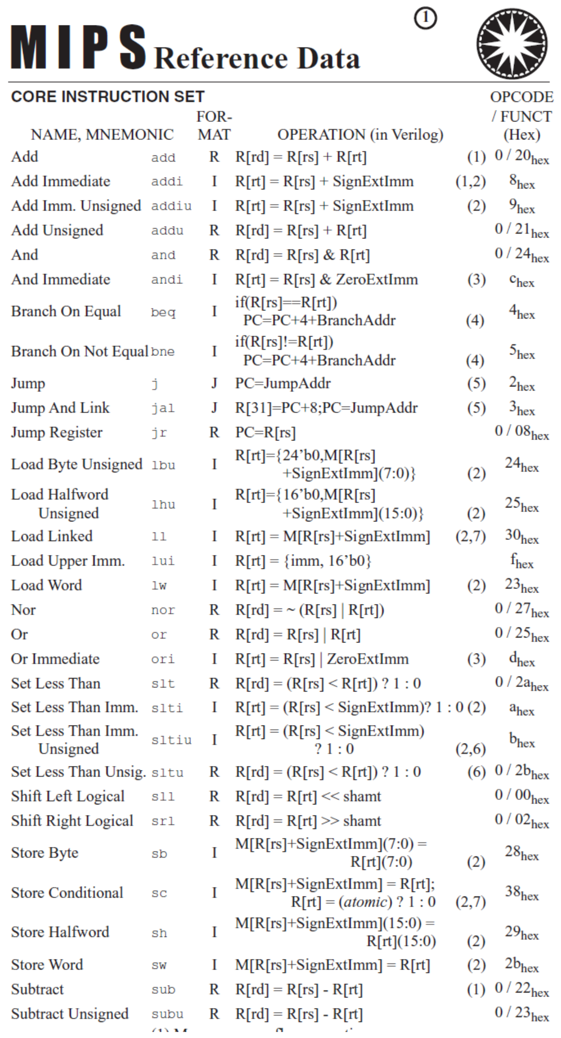
CSE 140 Lab/HW#3 – Due: in 2 weeks (see below)

MIPS Decoder (80pts)

Form a group of two members and do the following tasks.

* Note that this programming assignment is closely related to the project so work with the team member who you want to do the project together.

Write a C program (Decode.c) that decodes an input machine code. Your program needs to be able to decode the R, I, and J type instructions listed below. We will only test the following instructions. You can find these instructions from the first page of the MIPS Reference Data sheet (download from CatCourse).



MIPS Reference Data ①

CORE INSTRUCTION SET

NAME, MNEMONIC		FOR-MAT	OPERATION (in Verilog)		OPCODE / FUNCT (Hex)
Add	add	R	R[rd] = R[rs] + R[rt]	(1)	0 / 20_{hex}
Add Immediate	addi	I	R[rt] = R[rs] + SignExtImm	(1,2)	8_{hex}
Add Imm. Unsigned	addiu	I	R[rt] = R[rs] + SignExtImm	(2)	9_{hex}
Add Unsigned	addu	R	R[rd] = R[rs] + R[rt]		0 / 21_{hex}
And	and	R	R[rd] = R[rs] & R[rt]		0 / 24_{hex}
And Immediate	andi	I	R[rt] = R[rs] & ZeroExtImm	(3)	c_{hex}
Branch On Equal	beq	I	if(R[rs]==R[rt]) PC=PC+4+BranchAddr	(4)	4_{hex}
Branch On Not Equal	bne	I	if(R[rs]!=R[rt]) PC=PC+4+BranchAddr	(4)	5_{hex}
Jump	j	J	PC=JumpAddr	(5)	2_{hex}
Jump And Link	jal	J	R[31]=PC+8;PC=JumpAddr	(5)	3_{hex}
Jump Register	jr	R	PC=R[rs]		0 / 08_{hex}
Load Byte Unsigned	lbu	I	R[rt]={24'b0,M[R[rs] +SignExtImm](7:0)}	(2)	24_{hex}
Load Halfword Unsigned	lhu	I	R[rt]={16'b0,M[R[rs] +SignExtImm](15:0)}	(2)	25_{hex}
Load Linked	ll	I	R[rt] = M[R[rs]+SignExtImm]	(2,7)	30_{hex}
Load Upper Imm.	lui	I	R[rt] = {imm, 16'b0}		f_{hex}
Load Word	lw	I	R[rt] = M[R[rs]+SignExtImm]	(2)	23_{hex}
Nor	nor	R	R[rd] = ~ (R[rs] | R[rt])		0 / 27_{hex}
Or	or	R	R[rd] = R[rs] | R[rt]		0 / 25_{hex}
Or Immediate	ori	I	R[rt] = R[rs] | ZeroExtImm	(3)	d_{hex}
Set Less Than	slt	R	R[rd] = (R[rs] < R[rt]) ? 1 : 0		0 / $2a_{hex}$
Set Less Than Imm.	slti	I	R[rt] = (R[rs] < SignExtImm)? 1 : 0	(2)	a_{hex}
Set Less Than Imm. Unsigned	sltiu	I	R[rt] = (R[rs] < SignExtImm) ? 1 : 0	(2,6)	b_{hex}
Set Less Than Unsig.	sltu	R	R[rd] = (R[rs] < R[rt]) ? 1 : 0	(6)	0 / $2b_{hex}$
Shift Left Logical	sll	R	R[rd] = R[rt] << shamt		0 / 00_{hex}
Shift Right Logical	srl	R	R[rd] = R[rt] >> shamt		0 / 02_{hex}
Store Byte	sb	I	M[R[rs]+SignExtImm](7:0) = R[rt](7:0)	(2)	28_{hex}
Store Conditional	sc	I	M[R[rs]+SignExtImm] = R[rt]; R[rt] = (*atomic*) ? 1 : 0	(2,7)	38_{hex}
Store Halfword	sh	I	M[R[rs]+SignExtImm](15:0) = R[rt](15:0)	(2)	29_{hex}
Store Word	sw	I	M[R[rs]+SignExtImm] = R[rt]	(2)	$2b_{hex}$
Subtract	sub	R	R[rd] = R[rs] - R[rt]	(1)	0 / 22_{hex}
Subtract Unsigned	subu	R	R[rd] = R[rs] - R[rt]		0 / 23_{hex}

When the program is started, your program will print “Enter an instruction in machine code:” and receive one 32-bit machine code from user. Then, the program will decode the machine code and print the information (type and values of individual fields of the instruction format) of the instruction. The followings are the example execution of the program. Your program must produce the following sample results in the exact same format.

Sample results (inputs are presented in italic and bold font):

```
Enter an instruction in machine code:

00000001000010011000100000100000

Instruction Type: R
Operation: add
Rs: t0 (R8)
Rt: t1 (R9)
Rd: s1 (R17)
Shamt: 0
Funct: 32


Enter an instruction:

00110100001001000000000000000000

Instruction Type : I
Operation: ori
Rs: at (R1)
Rt: a0 (R4)
Immediate: 0
```

As a group, discuss how to read and interpret the instructions from user. Once this step is done, split the task among your group so that each member will implement different instructions individually.

Assume that your program will always receive machine codes in correct formats. So, you do not need to implement invalid input handling function. All test cases will follow the correct machine code format.

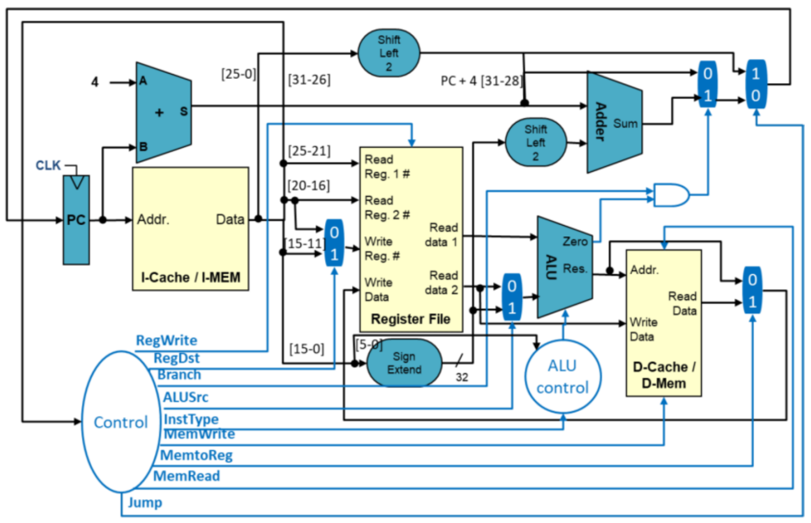
Demo : You will have two weeks to finish this lab. However, as an intermediate check, you need to show a demo to your TA one instruction per type (R, I, and J) at the next lab time. Therefore, your program needs to be able to decode at least one instruction per instruction type by next lab. And then you will complete the program and submit the final program by the deadline. When demoing and submitting your program, indicate which portion of the code was your implementation. “Working on the code together” means someone did not implement individually.

Single-cycle MIPS Architecture (10pts)

1. Assume that core components of single-cycle processor (shown below) have the following latencies:

I-Mem	Adder	Mux	ALU	Regs Rd/Wr	D-Mem	Sign-Extend	Shift-Left-2
40ps	50ps	20ps	100ps	80ps/60ps	200ps	20ps	20ps

Single-cycle processor data path:



Suppose that this data path executes only two types of instruction:

```
sub $rd, $rs, $rt

lw  $rt, offset($rs)
```

What would be the clock period to correctly execute the two instructions on the above single-cycle processor? Assume that PC register doesn't take any latency (i.e. Propagating a new PC value to I-Cache/I-Mem doesn't take any cycle).

Single and Pipelined Datapaths (10pts)

2. Assume that the execution time of individual steps of an instruction execution are like below:

IF	ID	EX	MEM	WB
100ps	120ps	220ps	300ps	120ps

a. If you design a single-cycle processor with the above latency information, what is the clock latency?

b. If you design a five-stage pipelined processor with the above latency information, what is the clock latency?

Submission Guideline

- Submit the Decode.c code named “first-member-name_second-member-name.c”, and your solution for the single-cycle and pipelined datapath problems in a separate MS Word or a pdf format to the CatCourse.
- Deadline: **11:59PM of one day before the lab in two weeks** (If this lab is assigned on 2/16, the deadline is 3/1 11:59PM)