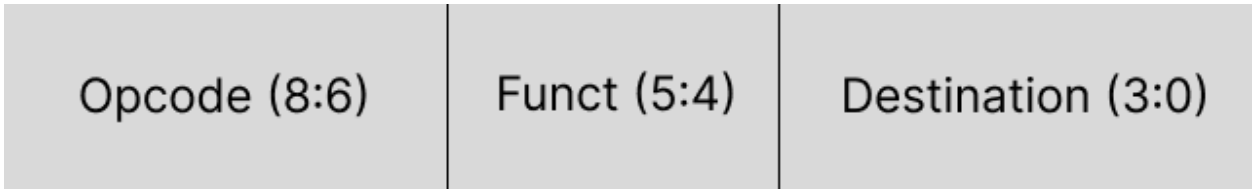
DNAM (**D**efinitely **N**ot **A**RM or **M**IPS)
Reference Sheet / README

Inspired by: https://github.com/yehzhang/x9

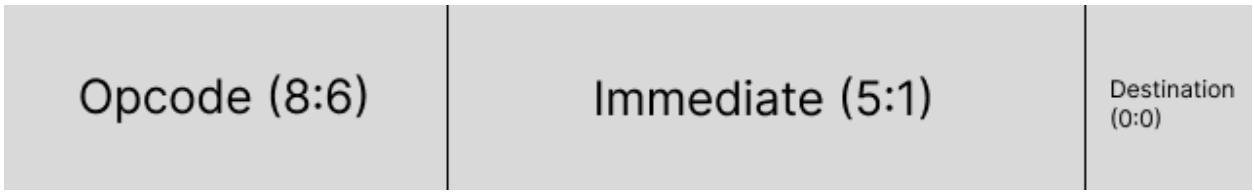
Instr	**Type**	**What it does**	**Usage**	**Opcode**	**Funct**	**Notes**
add	R	{sc_o, R[rd]} = R[0] + R[1] + sc_i	add rd	000	00	Clears parity bit (as does sub)
sub	R	{sc_o, R[rd]} = R[0] - R[1] + sc_i	sub rd	000	01	
lbr	R	R[rd] = Mem[R[0]]	lbr rd	000	10	
sbr	R	Mem[R[0]] = R[rd]	sbr rd	000	11	
lb	I	R[rt] = Mem[mem_lut[imm]]	lb rt, imm	001	-	rt can be either r0 or r1
subi	I	{sc_o, R[rt]} = R[0] - Mem[alu_lut[imm]] + sc_i	subi rt, imm	010	-	
addi	I	{sc_o, R[rt]} = R[0] + Mem[alu_lut[imm]] + sc_i	addi rt, imm	011	-	
beq	B	if (R[0] == R[1]) pc = PC_lut[imm]	beq imm	100	00	
bne	B	if (R[0] != R[1]) pc = PC_lut[imm]	bne imm	100	01	

blt	B	if (R[0] < R[1]) pc = PC_lut[imm]	blt imm	100	10	
ble	B	if (R[0] < R[1]) pc = PC_lut[imm]	ble imm	100	11	
mov	M	R[rt] = R[rs]	mov rt, rs	101	0	rt can be either r0 or r1, while rs can be any register
mov	M	R[rs] = R[rt]	mov rs, rt	101	1	
lsl	R	{sc_o, R[rd]} = {R[0], sc_i}	lsl rd	110	00	Clears parity bit (as do other shifts)
asr	R	{R[rd], sc_o} = {R[0][7], R[0]}	asr rd	110	01	
lsr	R	{R[rd], sc_o} = {sc_i, R[0]}	lsr rd	110	10	
not	R	R[rd] = !R[0]	not rd	110	11	Clears carry bit (as do other bitwise ops)
and	R	R[rd] = R[0] & R[1]	and rd	111	00	
xor	R	R[rd] = R[0] ^ r[1]	xor rd	111	01	
rxor	R	R[rd] = ^R[0] ^ pari	rxor rd	111	10	Modifies parity bit
or	R	R[rd] = R[0] | R[1]	or rd	111	11	

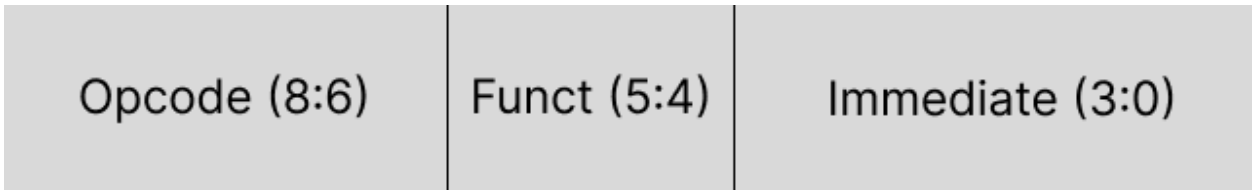
R Type Instruction Format*



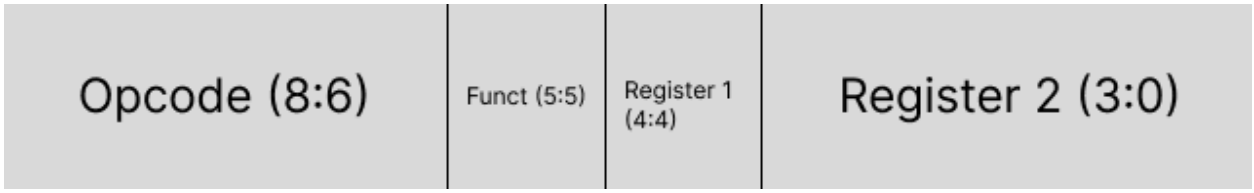
I Type Instruction Format



B Type Instruction Format



M Type Instruction Format



* Diagrams are not to scale

Assembler Usage

./assembler <assembly code> <machine code>

Video Link / Passcode

https://ucsd.zoom.us/rec/share/9nBqjCIR4Hbk6pVVuWx6KwgxSIbsPg8MVRLLUtCXNGvwCP-1yYwOfYF-qYoT568M.ww_dgaB0gQUGMRUT
Passcode: 21zYug$8
(view recording number 4! It is the only good one)