CS2100 Quick Reference Sheet

General Purpose Registers

Name	Reg. Number
\$zero	0
\$v0-\$v1	2-3
\$a0-\$a3	4-7
\$t0-\$t7	8-15
\$s0-\$s7	16-23
\$t8-\$t9	24-25
\$gp	28
\$sp	29
\$fp	30
\$ra	31

Opcode/Funct Reference

	Opcode	Funct
add	000 000	100 000
sll	000 000	000 000
srl	000 000	000 010
slt	10 10 10	
slti	00 10 10	
beq	000 100	
bne	000 101	
j	000 010	

R-format

arith \$rd, \$rs, \$rt   **shift** \$rd, \$rt, shamt.   **(\$rs=0)!!!!**

opcode [6]	rs [5]	rt [5]	rd [5]	shamt [5]	funct [6]

I-format

- arith \$rt, \$rs, $C16_{2s}$
- ld/st \$rt, $C16_{2s}$(\$rs)
- logic \$rt, \$rs, C16
- branch \$rs, \$rt, label

opcode [6]	rs [5]	rt [5]	Immediate [16]

J-format

opcode [6]	target address [first 4 MSB is PC+4] [26]

Logic:

a	b	AND	OR	NOR	XOR
0	0	0	0	1	0
0	1	0	1	0	1
1	0	0	1	0	1
1	1	1	1	0	0

IEEE 754 Floating-Point Rep

sign [1]	exponent [8]	mantissa [23]

Power of 2 Table

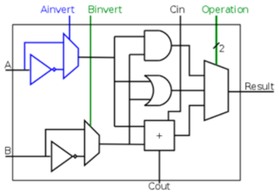
Exp	Val	Exp	Val	Exp	Val	Exp	Val
2^0	1	2^8	256	2^{16}	65,536	2^{24}	16,777,216
2^1	2	2^9	512	2^{17}	131,072	2^{25}	33,554,432
2^2	4	2^{10}	1,024	2^{18}	262,144	2^{26}	67,108,864
2^3	8	2^{11}	2,048	2^{19}	524,288	2^{27}	134,217,728
2^4	16	2^{12}	4,096	2^{20}	1,048,576	2^{28}	268,435,456
2^5	32	2^{13}	8,192	2^{21}	2,097,152	2^{29}	536,870,912
2^6	64	2^{14}	16,384	2^{22}	4,194,304	2^{30}	1,073,741,824
2^7	128	2^{15}	32,768	2^{23}	8,388,608	2^{31}	2,147,483,648

Exp	Val	Exp	Val
2^{-1}	0.5	2^{-9}	0.001953125
2^{-2}	0.25	2^{-10}	0.0009765625
2^{-3}	0.125	2^{-11}	0.00048828125
2^{-4}	0.0625	2^{-12}	0.000244140625
2^{-5}	0.03125	2^{-13}	0.0001220703125
2^{-6}	0.015625	2^{-14}	0.00006103515625
2^{-7}	0.0078125	2^{-15}	0.000030517578125
2^{-8}	0.00390625	2^{-16}	0.0000152587890625

# of Digits	Radix	(b-1)s	(b)s
n	**b**	$-x = b^n - x - 1$	$-x = b^n - x$

Control Signal

	RegDst	RegWrite	ALUSrc	ALUcontrol/op	MemRead	MemWrite	MemToReg	Branch
R	1	1	0	?/?	0	0	0	0
lw	0	1	1	0010/00	1	0	1	0
sw	X	0	1	0010/00	0	1	X	0
beq	X	0	0	0110/01	0	0	X	1



ALUcontrol			Function
Ainvert	Binvert	Operation	
0	0	00	AND
0	0	01	OR
0	0	10	add
0	1	10	subtract
0	1	11	slt
1	1	00	NOR

Laws of Boolean Algebra

Identity laws	
A + 0 = 0 + A = A	A · 1 = 1 · A = A
Inverse/complement laws	
A + A' = A' + A = 1	A · A' = A' · A = 0
Commutative laws	
A + B = B + A	A · B = B · A
Associative laws *	
A + (B + C) = (A + B) + C	A · (B · C) = (A · B) · C
Distributive laws	
A · (B + C) = (A · B) + (A · C)	A + (B · C) = (A + B) · (A + C)

Idempotency	
X + X = X	X · X = X
One element / Zero element	
X + 1 = 1 + X = 1	X · 0 = 0 · X = 0
Involution	
(X')' = X	
Absorption 1	
X + X·Y = X	X·(X + Y) = X
Absorption 2	
X + X'·Y = X + Y	X·(X' + Y) = X·Y
De Morgans' (can be generalised to more than 2 variables)	
(X + Y)' = X' · Y'	(X · Y)' = X' + Y'
Consensus	
X·Y + X'·Z + Y·Z = X·Y + X'·Z	(X+Y)·(X'+Z)·(Y+Z) = (X+Y)·(X'+Z)

Flip-flop Characteristic Tables

J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	Q(t)'

S	R	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	?

D	Q(t+1)
0	0
1	1

T	Q(t+1)
0	Q(t)
1	Q(t)'

Flip-flop Excitation Tables

Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

JK Flip-flop

Q	Q^+	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

SR Flip-flop

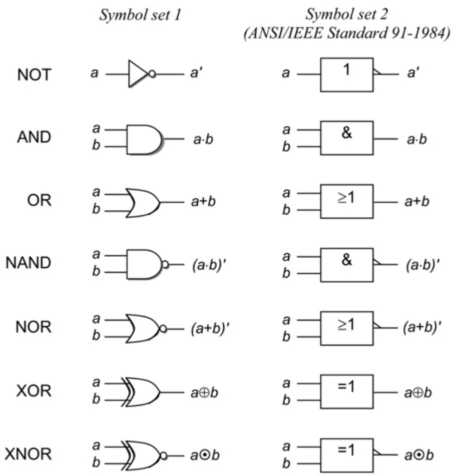
Q	Q^+	D
0	0	0
0	1	1
1	0	0
1	1	1

D Flip-flop

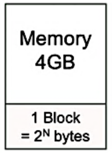
Q	Q^+	T
0	0	0
0	1	1
1	0	1
1	1	0

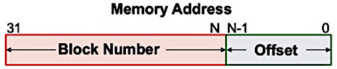
T Flip-flop

Circuit Table

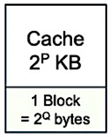


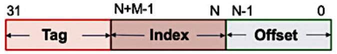
Cache





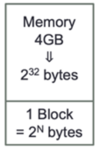
Offset: N bits
Block Number = 32 – N bits
Check: Number of Blocks = 2^{32-N}

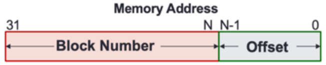




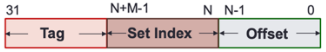
Offset N = Q bits
Number of Cache Blocks
= 2^P **KB /** 2^Q **bytes = (2^P / 2^Q) * 2^{10}**
Cache Index M = P – Q + 10
Cache Tag = 32 – Index - Offset

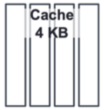
1Bytes=8bits
1KB=1024Bytes
1MB=1024KB
1GB=1024MB
1TB=1024GB





Offset = N bits
Block Number = 32 – N **bits**
Check: Number of Blocks = 2^{32-N}





Number of Cache Blocks
= 4KB / 4bytes = 1024 **=** 2^{10}

4-way associative, number of sets
= 1024 / 4 = 256 **=** 2^8
Set Index, M = 8 bits

Cache Tag = 32 – 8 – 2 = **22 bits**

Pipeline [Ideal: $I + N - 1$]

Without Forwarding (Assume No Early Branching)

Non-LW (+2)									
add	F	D	E	M	W				
sub		F	🟨	🟨	D	E	M	W	

LW (+2)									
lw	F	D	E	M	W				
sub		F	🟨	🟨	D	E	M	W	

Non-LW before Branch (+2)									
add	F	D	E	M	W				
beq		F	🟨	🟨	D	E	M	W	

LW before Branch (+2)									
add	F	D	E	M	W				
beq		F	🟨	🟨	D	E	M	W	

After Branch (+3)									
beq	F	D	E	M	W				
add		🟨	🟨	🟨	F	D	E	M	W

With Forwarding but No Early Branching

Non-LW (+0)									
add	F	D	E	M	W				
sub		F	D	E	M	W			

LW (+1)									
lw	F	D	E	M	W				
sub		F	D	🟨	E	M	W		

Non-LW before Branch (+0)									
add	F	D	E	M	W				
beq		F	D	E	M	W			

LW before Branch (+1)									
add	F	D	E	M	W				
beq		F	D	🟨	E	M	W		

After Branch (+3)									
beq	F	D	E	M	W				
add		🟨	🟨	🟨	F	D	E	M	W

Branch

Branch without Early Branch with Branch Prediction (Taken) (+3)									
beq	F	D	E	M	W				
inst1		F	D	E	M	W			
inst2			F	D	E	M	W		
inst3				F	D	E	M	W	
label		🟨	🟨	🟨	F	D	E	M	W

Branch with Early Branch with Branch Prediction (Taken) (+1)									
beq	F	D	E	M	W				
inst1		F	D	E	M	W			
label		🟨	F	D	E	M	W		

Branch with(out) Early Branch with Branch Prediction (Not Taken) (+0)									
beq	F	D	E	M	W				
inst1		F	D	E	M	W			
inst2			F	D	E	M	W		

With Forwarding and Early Branching

Non-LW (+0)									
add	F	D	E	M	W				
sub		F	D	E	M	W			

LW (+1)									
lw	F	D	E	M	W				
sub		F	D	🟨	E	M	W		

Non-LW before Branch (+1)									
add	F	D	E	M	W				
beq		F	🟨	D	E	M	W		

LW before Branch (+2)									
add	F	D	E	M	W				
beq		F	🟨	🟨	D	E	M	W	

After Branch (+1)									
beq	F	D	E	M	W				
add		🟨	F	D	E	M	W		