RISC V
ARM

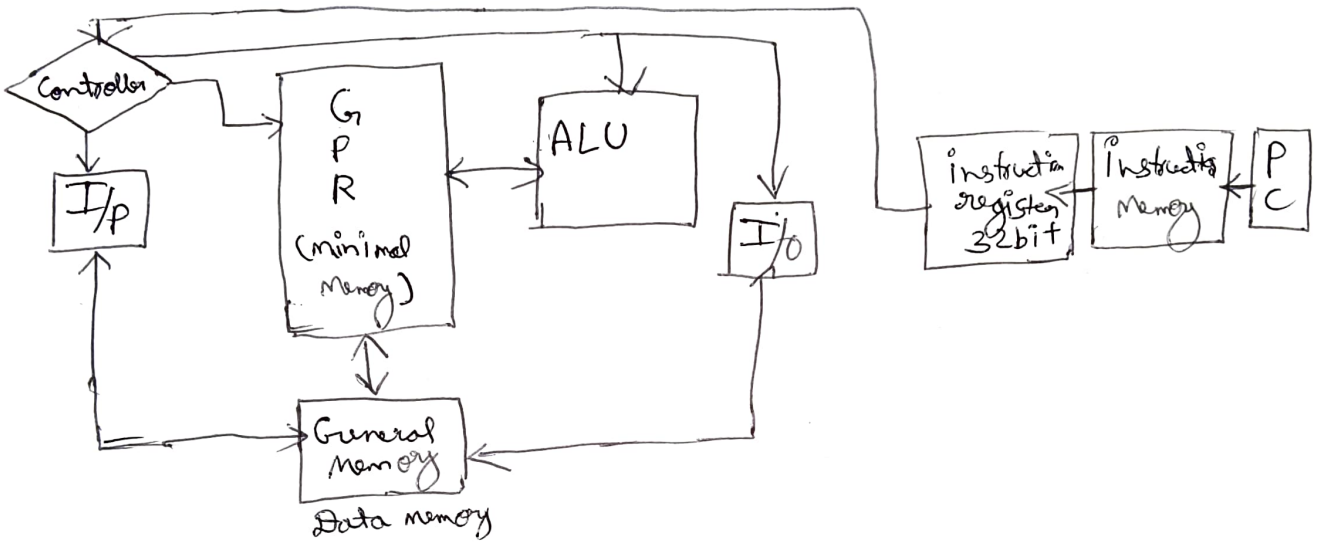
(ARM - LRM)
MIPS 32

CPU → Hardware → High level Programming → Compiler → Assembly language → Binary

a = b + c ⇒ HLP → compiler

$R_1 = a$ $R_2 = b$ $R_3 = c$
ADD R_1, R_2, R_3 } Assembly → assembler → Binary

Microarchitecture:-



GPR = General Purpose register
ALU = Arithmatic logical Unit
32 bit = instruction Register

PC (Program Controller)

Decoder ⇒ control signal (sel in MUX, ALU instruction etc.)

↓

Decode

↓

instruction Register

32 bit IR (SRAM)

0–31 (32 bit)

31:28	27:26	25:20	19–16	15:12	11:0
Cond	OP	Funct	Rn	Rd	SRC2
4	2	6	R_2 SRC1 4	4	12 enough

RISC

RISC V

ARM

(ARM - LRM)

MISP 32

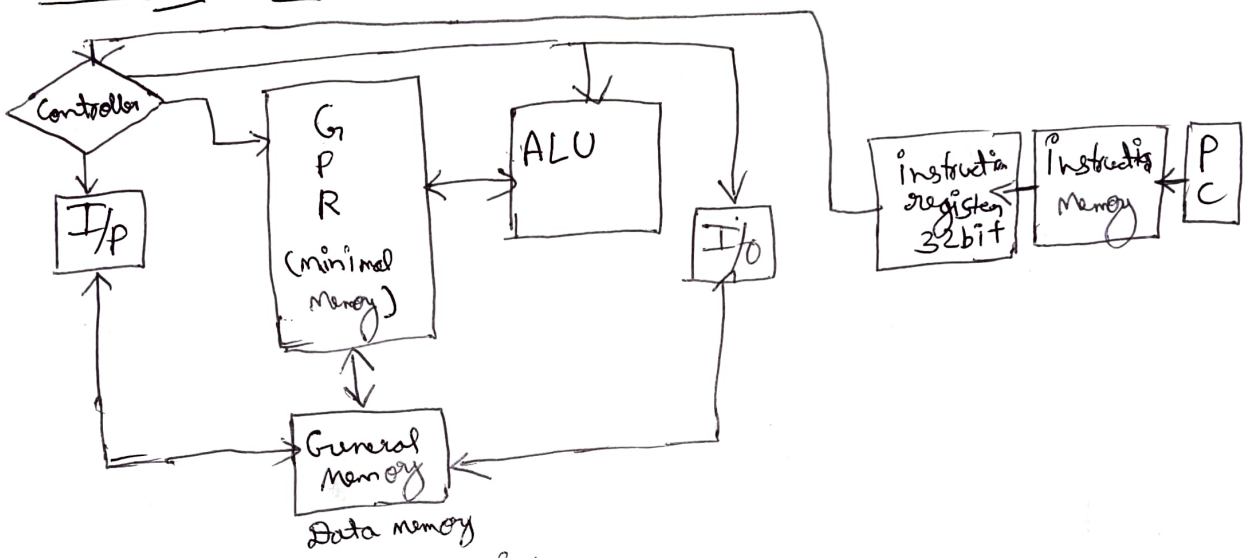
CPU → Hardware → High level Programing → Compiler → Assembly language → Binary

$a = b + c$ => HLP → compiler

$R_1 = a$ $R_2 = b$ $R_3 = c$

ADD R_1, R_2, R_3 } Assembly → assembler → Binary

Microarchitecture:-



GPR = General Purpose register

ALU = Arithmatic logical Unit

32 bit = instruction Register

PC (Program Controller)

Decoder => control signal (sel in MUX, ALU instruction etc.)

↓

Decode

↓

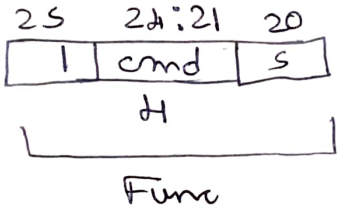
instruction Register

32 bit IR (SRAM)

0-31 (32 bit)

31:28	27:26	25:20	19-16	15:12	11:0
Cond	op	Funct	Rn	Rd	Src2
4	2	6	R_2 Src1	4	12
			4		4 enough

Func



$2^4 = 16$
Cond & op for more command

I ⇒ 1 = Immediate
0 = Register

ADD $R_1 R_2 R_3$ ⇒ Register
ADD $R_1, R_2, \#4$ ⇒ Immediate

everything in 32 bit
∴ databus is also be 32 bit
↳ To send 12 bit we will extend it (using signed bit)

Cond, S ⇒ comparision, overflow etc

CMP $R_1 R_2$
ADDEQ $R_3 R_1 R_2$ // If $(R_1 == R_2)$ $R_3 = R_1 + R_2$

Cond ⇒ EQ, NEQ etc 16 Condition as it is 4 bit sized

$2^4 = 16$

S = 1 will say condition flag is there or not (set or Notset)

SUBS $R_1 R_2 R_3$
ADD S R_1, R_2, R_3

Added Condition flag Considered
or else ignore.

OP

00 = Data Processing
01 = memory instruction
10 = Branch instruction

B = Branch
BL = Branch & link

Task

Data Processing

R → Src2 → Register ADD $R_1 R_2 R_3$

I → Src2 → Value ADD $R_1 R_2$ # 4

RISC (Fixed = 32bit)
CISC (Not fixed)

For I = 1 Immediate

11:8	7:0
rot	imm8

rot = rotation scheme
4 bits
0000
⋮
11110
0011

For I = 0 Registes

11:7	6:5	4	3:0
shamt5	sh	0	Rm

sh = shift
shamt5 = shift amount
5 = 32 bit = 2^5

LSL 00
LSR 01
ASR 10
ROR 11

destination Rd = First operator Register

Memory instruction

LDR
STR

31:28	27:28	25:20						19:16	15:12	11:0
Cond	01	I	P	U	B	W	L	Rn	Rd	src2

Rn ↓ Base

Base addressing :- R_{value} ± __ = Pre indexing

STR R_{11} = STR (R_3, # 4)

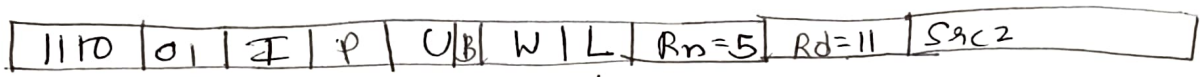
STR

U = Add/Subtract value for base register
1 = subtract
0 = Add

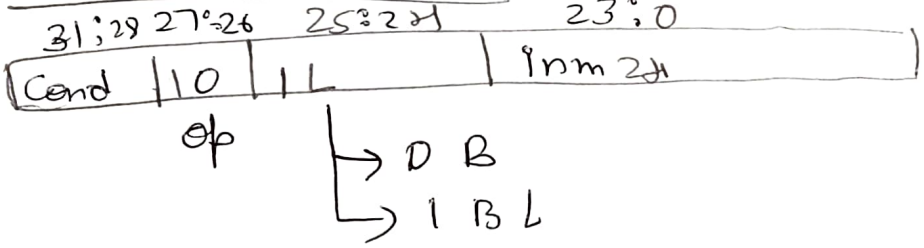
P	W	Index mode
0	0	Post-index
0	1	Not supported
1	0	offset
1	1	Pre index

L	B	inst
0	0	STR
0	1	STRB
1	0	LDR
1	1	LDRB

STR R11, [R5]#26

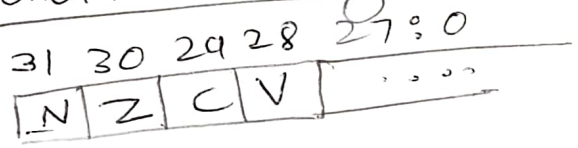


Branch Instruction



GS Jantung

Condition Flags (CPSR) => Current Program status register



Z = equal to

0000	EQ	equal	Z
0001	NE	Not equal	$\overline{Z}$
0010	CS/HS	Carry set/unsigned higher	C
0011	CC/LO	Carry clear/unsigned lower	$\overline{C}$
0100	MI	Minus/negative	N
0101	PL	Plus/positive/zero	$\overline{N}$
0110	VS	overflow set	V
0111	VC	overflow clear	$\overline{V}$
1000	HI	unsigned higher	$Z\overline{C}$
1001	LS	unsigned lower	$\overline{Z}C$
1010	GE	signed ≥	$\overline{N \oplus V}$
1011	LT	signed <	$N \oplus V$
1100	GT	signed >	$\overline{Z}(\overline{N \oplus V})$
1101	LE	signed ≤	$Z(N \oplus V)$

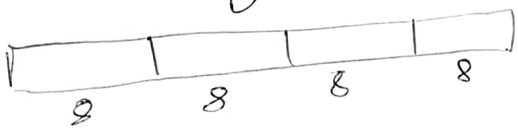
1110 AL Always (ignored)
(None)

Register file = 16 Register (32 bit) (GPR)

Data memory = 32×32 | 32 bit

2^{32} location

Instruction Memory = 32×32 2^{32} location



1 byte × 4

= 4 byte

⇓

0×00000000

0×00000004

⇓

PC ⇒ PC + 4

R_{15} → PC

read ↳ PC + 8 (due to pipelining)

Microarchitecture:

↳ single cycle

↳ multi cycle

↳ ~~AB~~ Pipelining

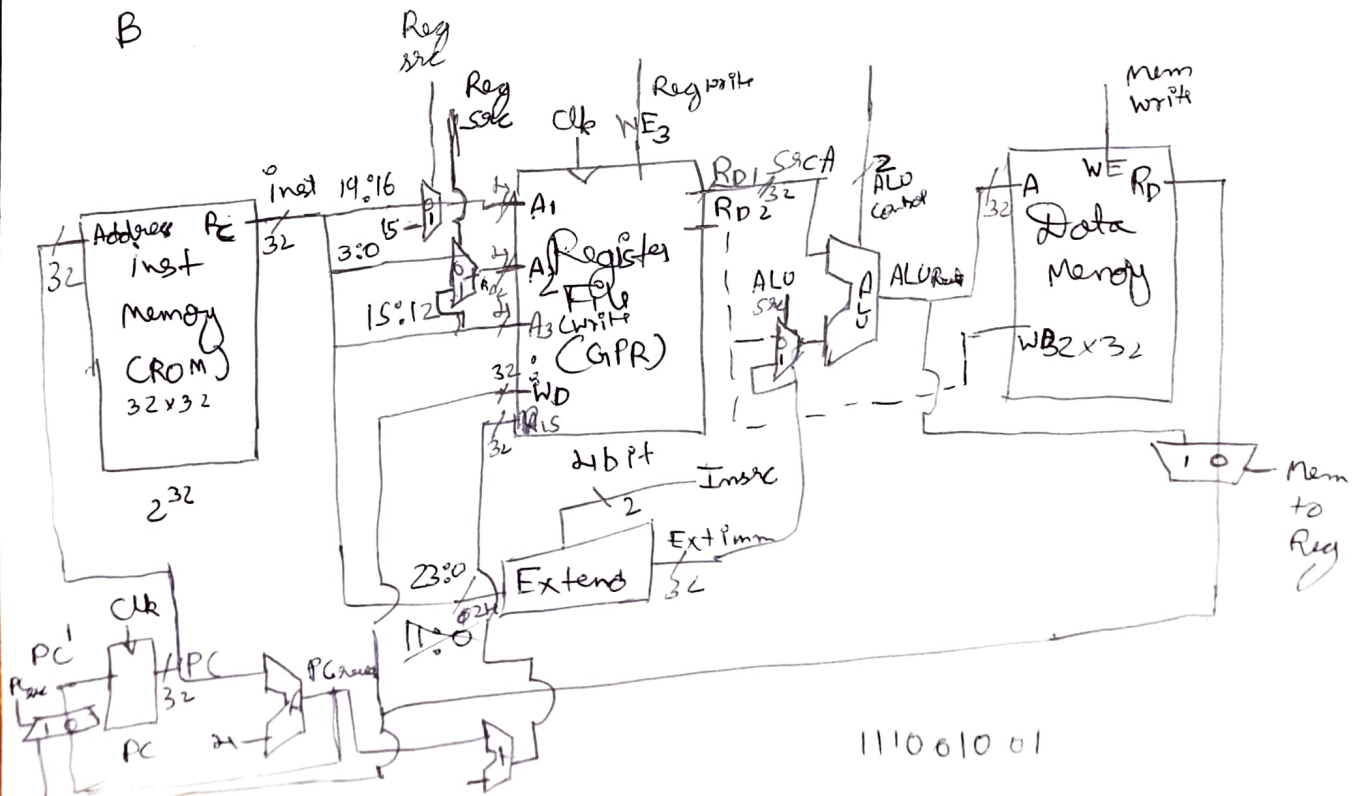
Instruction Set

ADD, SUB, AND, ORR

STR, LDR — = Bus

B

NE (control) = Write enable



Data Processing

ADD
AND
SUB
ORR

Imm src

0 -> ext 8 bit
1 -> ext 12 bit

Branch instruction

R_{15} + 24 bit from instruction

Imm src

00 ext 8 bit
01 ext 12 bit
10 ext 24 bit (signed)