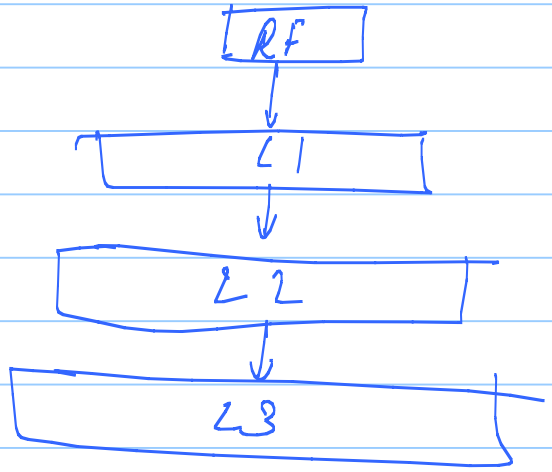
Aug. 9th

Note Title 09-08-2011

Physical View



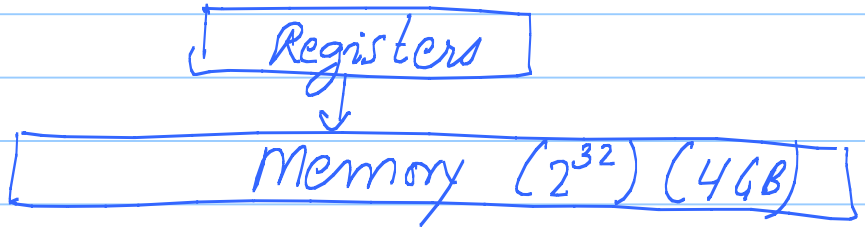
Logical View

→ Stack.

Software should be portable.

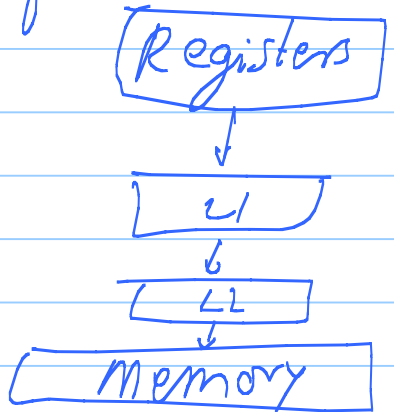
Compiler / Software View

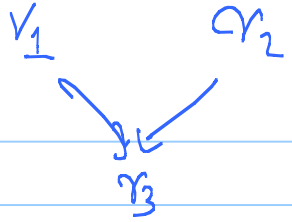
Architecture



Hardware's point of view

Organization

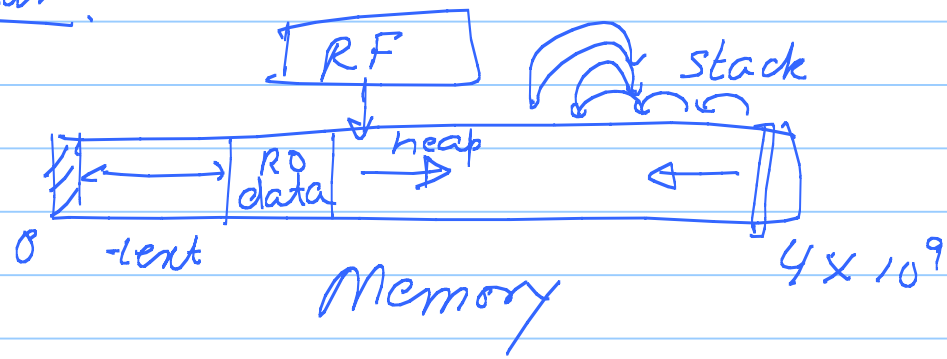




32×3

$96 \Longleftrightarrow 12$

Architecture



Simple Instructions

ADD r_1, r_2, r_3

$r_1 = r_2 + r_3$

SUB r_1, r_2, r_3

$r_1 = r_2 - r_3$

ADD $r_1, r_2, \#6$ ← immediate

$r_1 = r_2 + 6$

MUL r_1, r_2, r_3

$r_1 = r_2 \times r_3$

Logical Instructions.

ORR → OR instruction.
AND → AND
MVN → Not

LSL → Left Shift
LSR → Right Shift Logical.
ASR → Sign extend first bit

Data Transfer.

LDR → Load Register
STR → store Register.

base register

immediate offset

LDR r_5, [r_6, #10]

mem_address = r_6 + 10

Go to that address

fetch the data

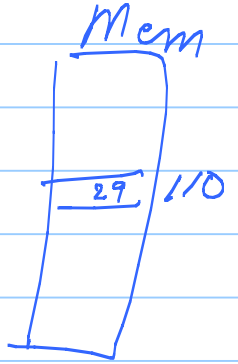
place it in r_5.

E.g.

r_6 = 100

r_6 + 10 = 110

r_5 = 29



LDR r_5, [r_6, r_7]

memory address is r_6 + r_7

LDR r_5, [r_6, r_7, LSL #2]

r_5 = mem [r_6 + (r_7 << 2)]

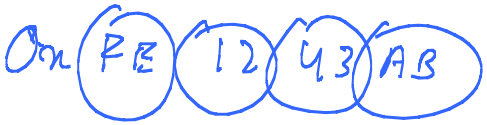
Array of integers: A[i]

r_6 → base address of A

r_7

char	short	int	long long int / double
(1)	(2)	(4)	(8)
LDRB	LDRH	LDR	(----) ?

How does LDR/STR work?

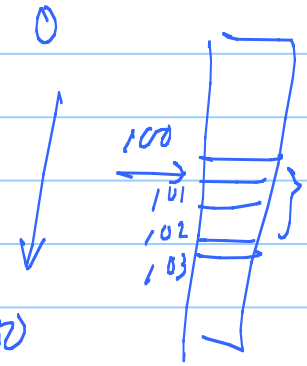


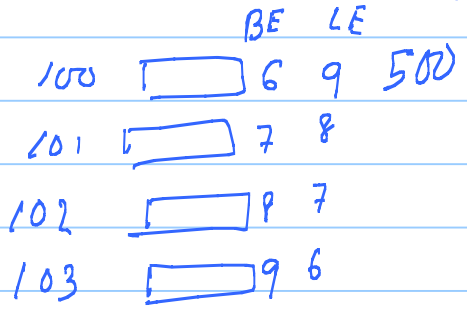
LDR r5, <mem_address>

100

Big Endian (Java) (SPARC, IBM PPC)

Little Endian. (ARM, x86 INTEL)





{ STMFD
{ LDMFD

ARM ARCHITECTURE REFERENCE MANUAL

$(2^{n-1}-1)$ $(-2^{(n-1)})$

Binary Numbers :

Two's Complement $(x) = 2^n - x$.

Robertson's Algorithm:

$x \times y$

$(-x) \times y = (2^n - x) \times y$

$= 2^n y - x \times y$

$(2^n - x)(2^n - y) = x \times y$

Q : 16^{bit} 32^{bit} Two's complement number.

Quick Way

$-x_{16}$: $2^{16} - x$

$-x_{32}$: $2^{32} - x$

} $2^{32} - 2^{16}$

Sign Extension: 0 → no problem

1 → extend for the left half.

Two's Complement

One's Complement

Biased.

$-5/2$: $2x - 3 + 1$

$2x - 2 - 1$

$127 + x$

} Rounding

3.5