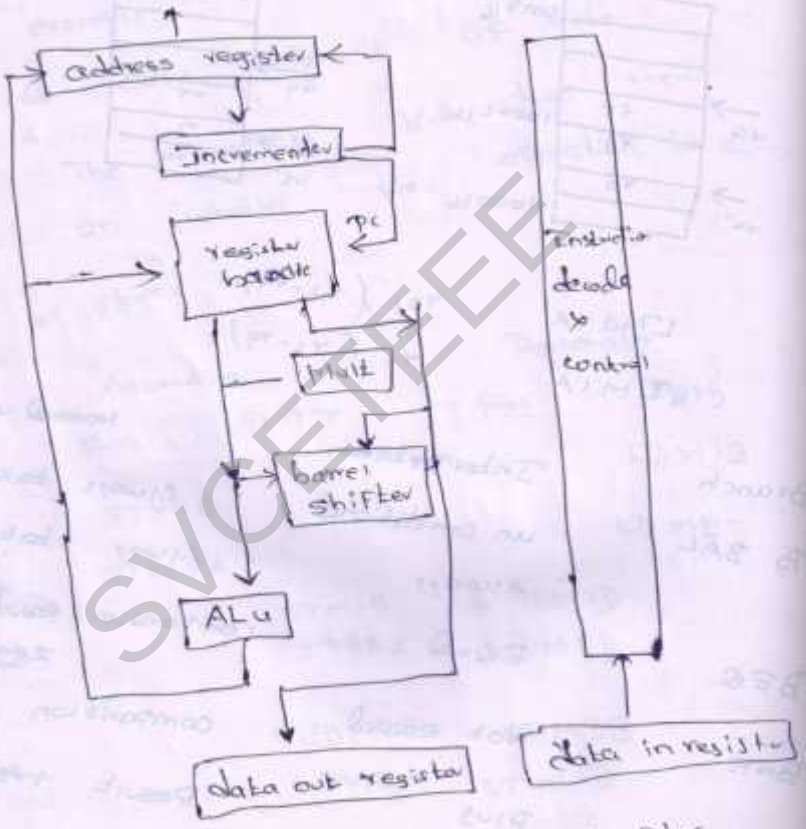
Unit - 5

ARM organization:-

3 stage pipe line ARM organization

The organization of an ARM with a 3 stage pipeline



The register bank which store the Processor state. It has 2 read ports & one write port which can be used to access any register

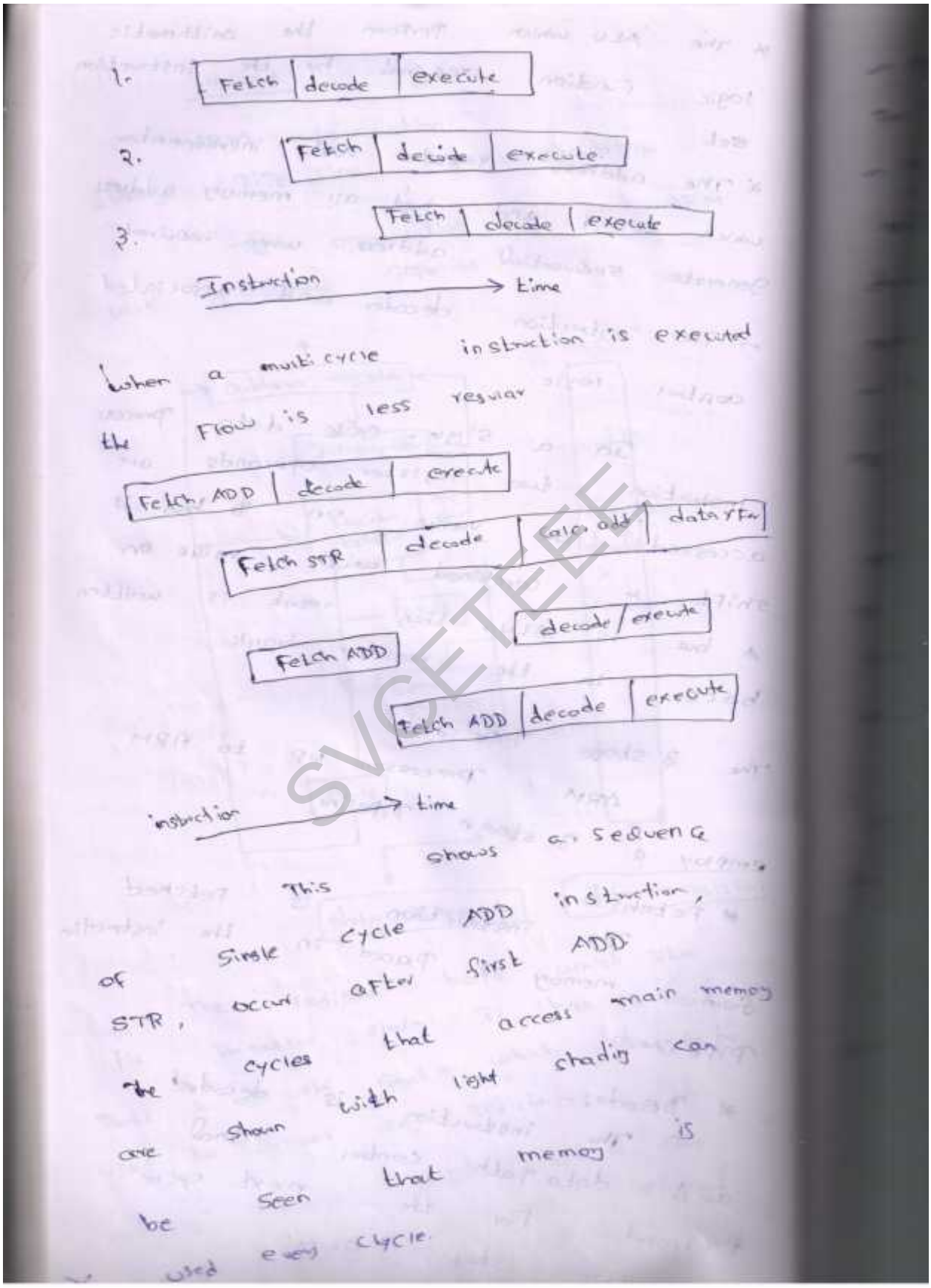
The barrel shifts which can rotate one operand or

* The ALU which perform the arithmetic logic function required by the instruction set
* The address register and incrementer which select and hold all memory address generate sequential address when required
→ The instruction decoder and associated control logic.

In a single-cycle data processing instruction, two register operands are accessed the value on B bus is shifted & combined with value on A bus is ALU. Then result is written back in the register bank.

The 3 stage pipeline:
ARM Processor up to ARM employ a 3 stage pipeline.

* Fetch:-
The instruction is fetched from memory & placed in the instruction pipeline
* Decode:-
The instruction is decoded and data path control signal prepared For the next cycle


1.
Fetch
decode
execute
2.
Fetch
decode
execute
3.
Fetch
decode
execute
Instruction
time
when a multi cycle instruction is executed
the Flow is less regular
Fetch ADD
decode
execute
Fetch STR
decode
calc. add
data xfer
Fetch ADD
decode/execute
Fetch ADD
decode
execute
instruction
time
This shows a sequence
of Single cycle ADD instruction,
STR, occur after first ADD
be cycles that access main memory
are shown with light shading can
be seen that memory is
used every cycle.

→ All instruction occupy the data path for one or more adjacent cycle

→ For each cycle that an instruction occupy the data path. it occupy the decode logic in immediate preceding cycle

→ Branch instruction flush & refill

PC Behaviour:-

one consequence of pipe lined executed model used on RAM is that program counter, which is visible to user r15, must run ahead of current instruction.

and the programmer who attempts to do directly this is indeed what happen through r15 must take. account such we of PC & unpredictable and it should be avoided. especially since later donot have same behaviour in these cases.

5 STAGE pipeline ARM organisation:-

$T_{prog} = N_{inst} \times CPI$

(a) Memory Bottle neck:-

Fundamental Problem with reducing the CPI relative to a 3 stage core is related to von neuman to data memory will have its Performance limit by the available memory band width. A 3 stage ARM core access memory on every clock cycle.

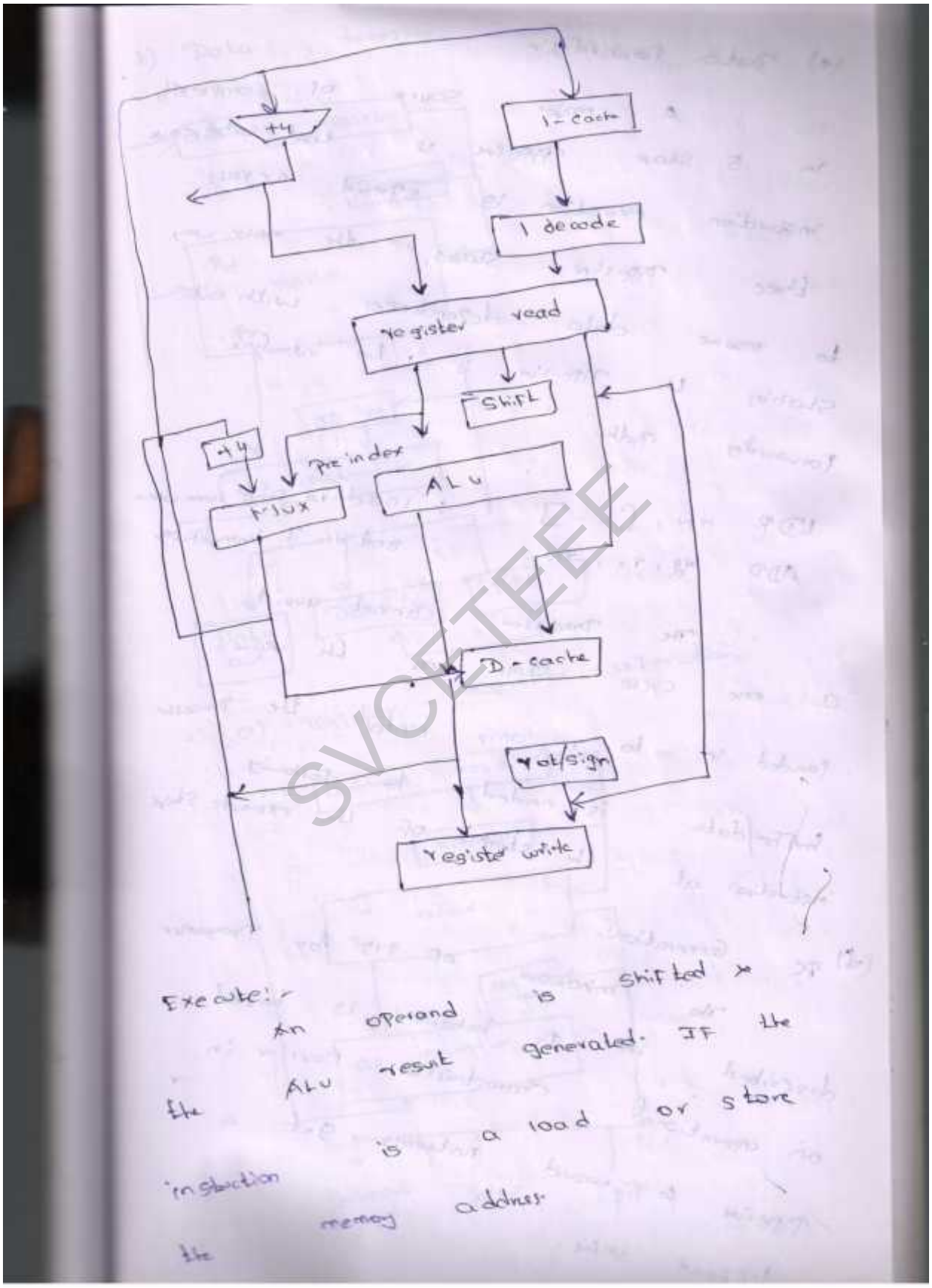
As the result of the above issues, the higher performance ARM cores employee a 5 stage pipeline and have seperate instruction & data memory

(b) The 5 stage Pipeline :-

Fetch:- The instruction Fetch from memory

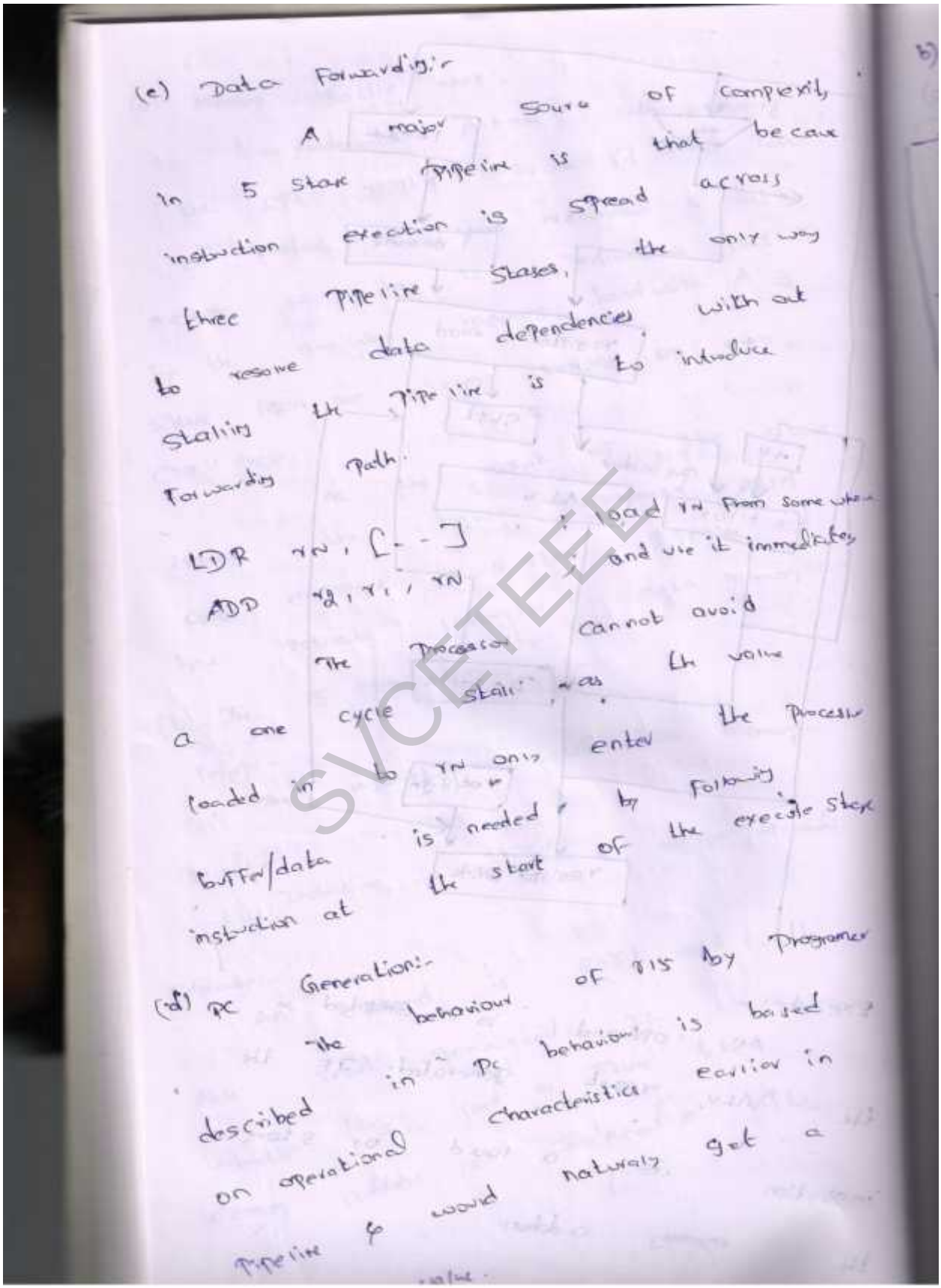
Decode:- Instruction is decoded & register operands

Execute:- An operand is shift and the ALU result generated IF the instruction is load or store the memory address is computed & ALU



Execute:-

An operand is shifted * the ALU result generated. If the instruction is a load or store the memory address



(e) Data Forwarding:-

A major source of complexity in 5 stage pipeline is that because instruction execution is spread across three pipeline stages, the only way to resolve data dependencies without stalling the pipeline is to introduce forwarding path.

LDR rN, [- -] ; load rN from some where

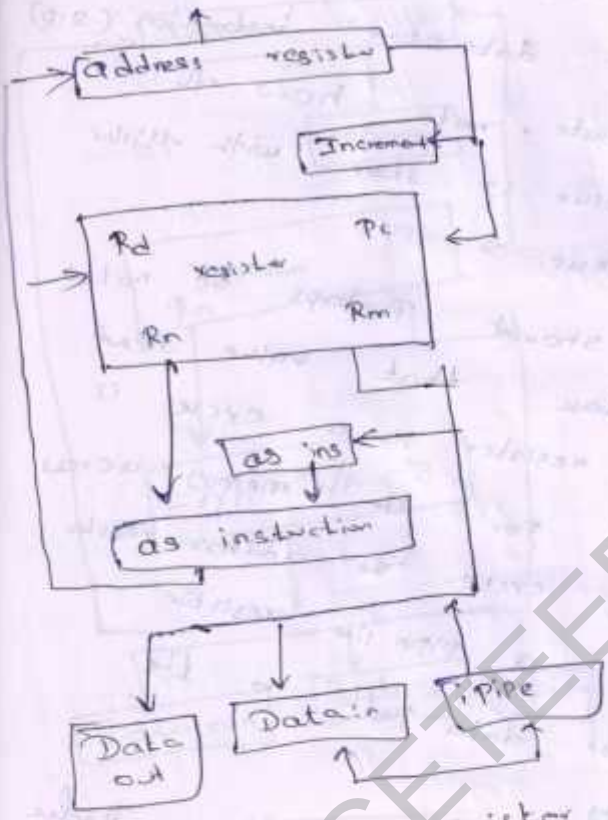
ADD r2, r1, rN ; and use it immediately

The Processor cannot avoid a one cycle stall as the value loaded in to rN only enter the Processor buffer/data is needed by following instruction at the start of the execute stage

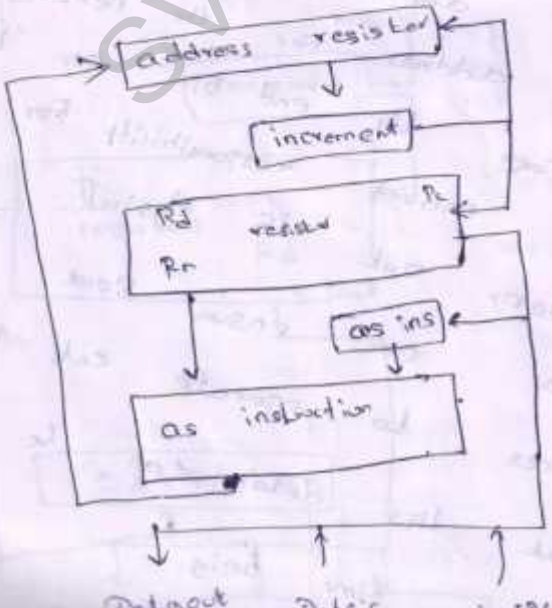
(f) PC Generation:-

The behaviour of PIS by Programmer described in PC behaviour is based on operational characteristics earlier in pipeline & would naturally get a

b) Data transfer Instructio



(a) register – register operation



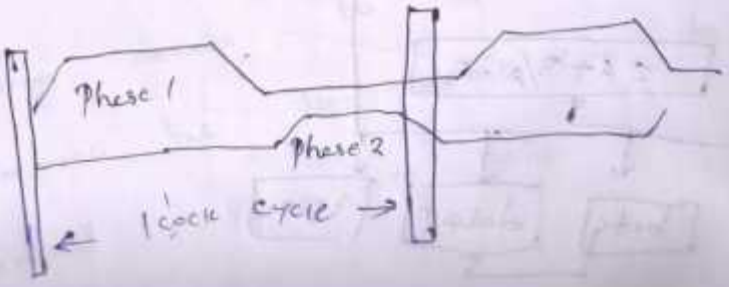
ARM Implementation:-

The Arm implementation follow a similar approach to that for mu0. the design is divided in to a data path section that is described in temalir

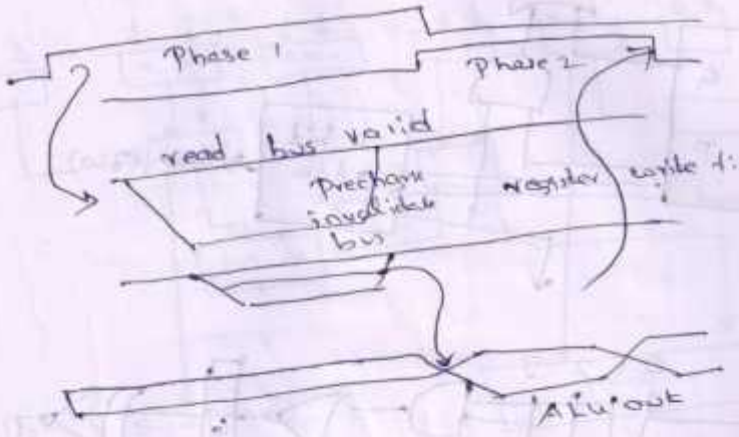
(a) Clocking scheme:-

most ARms donot operate with edge sensitive registers instead the design is based on 2 phase non overlap clock which are generate internally from a signal i/p clock

This scheme allows the use level sensitive transperent latches. Data movement is controlled by passing the 'data alternatively through latches which are open during phase 1 and which are open during phase 2. latches



(B) Data path Timing:



→ The minimum data path cycle time is there fore the sum of

*) register read time

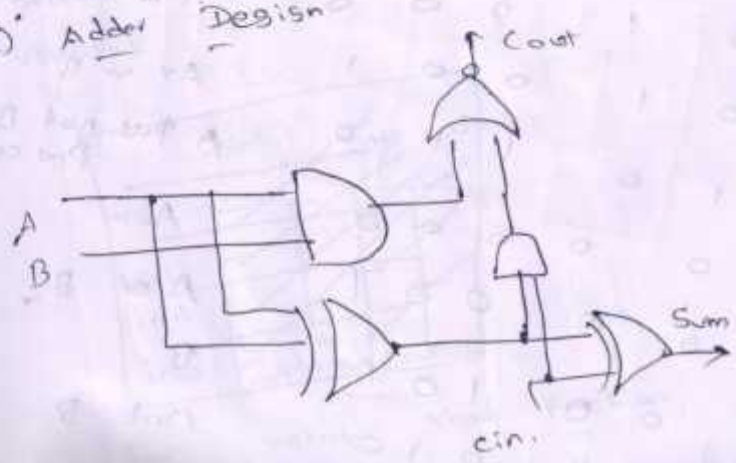
→ shift delay

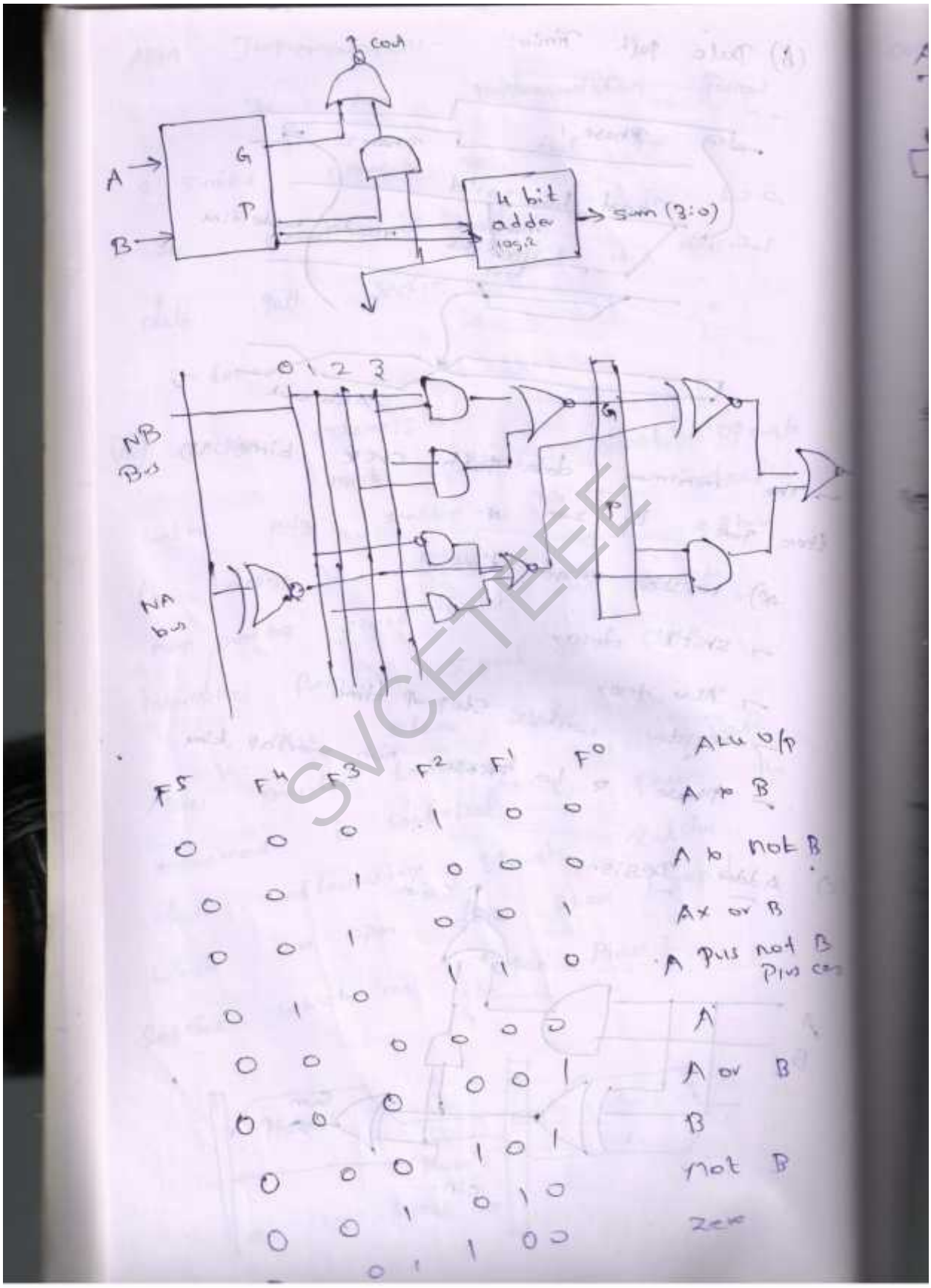
→ ALU delay

→ register write setup time

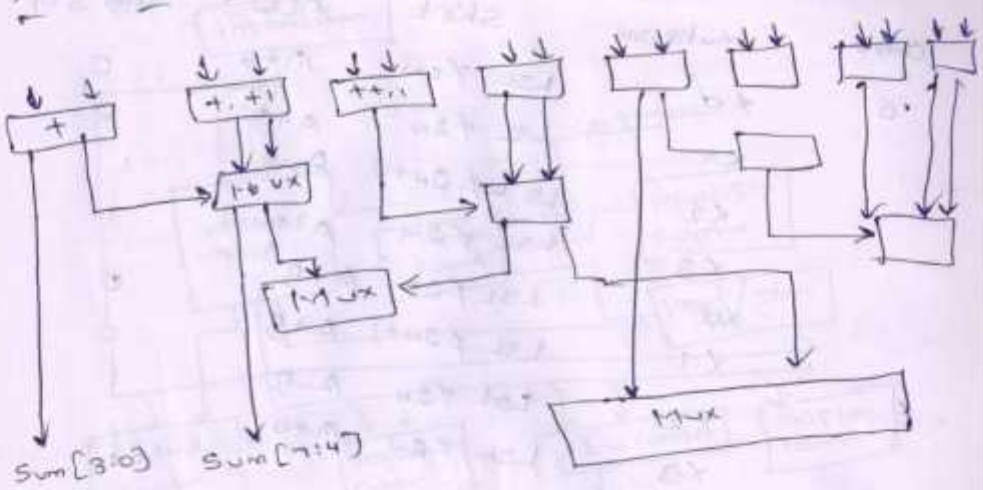
→ Phase 2 to phase 1 non overlap time

(C) Adder Design




cout
A
B
G
P
4 bit adder
sum (3:0)
0 1 2 3
NB Bus
NA bus
F5 F4 F3 F2 F1 F0
ALU o/p
A or B
A & not B
A ex or B
A plus not B Plus cin
A'
A or B
B
not B
Zero

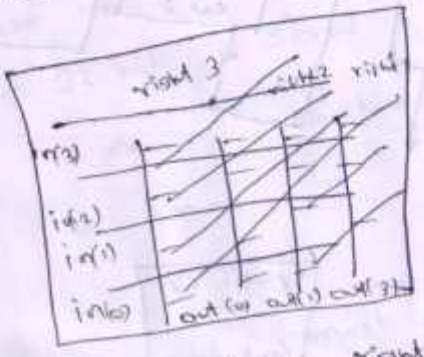
ARM 6 - carry select adder:



(9) carry Arbitration Adder:

$(u,v) * (u', v') = (v + u * u', v + u * v')$

The adder logic uses further improved on ARM and MI, where carry adder is used the carry arbitration scheme the conventional Proposed two reduce information in term of generate Variable

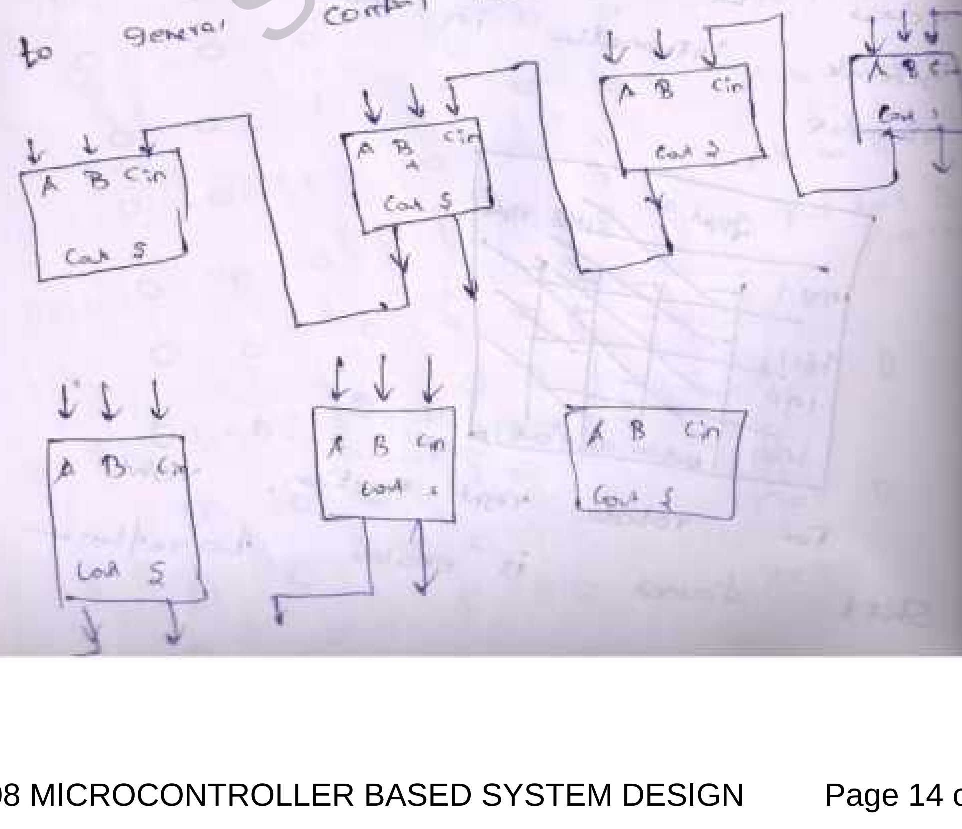


For rotate right function, right shift distance is enable together

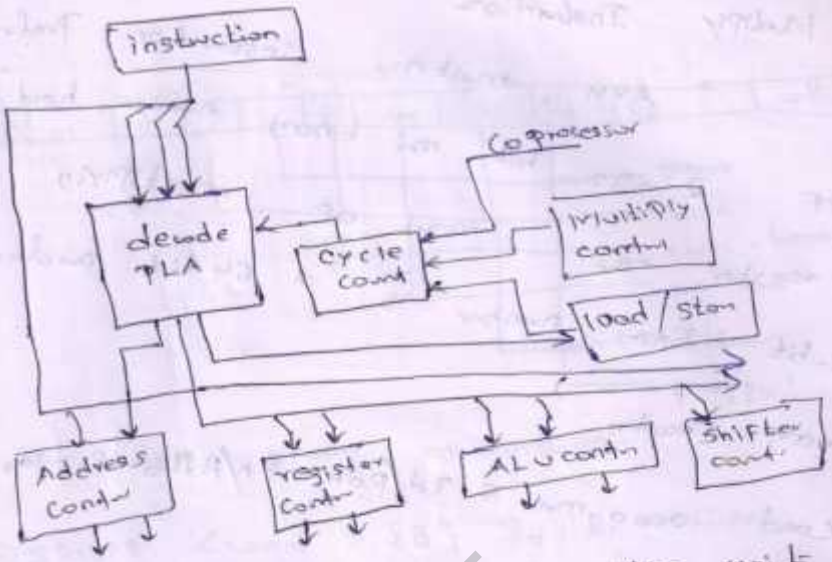
Carry in	multiplier	Shift	ALU	Carry out
0	x0	LSL #2N	A+0	0
	x1	LSL #2N	A+B	0
	x2	LSL #(2N+1)	A-B	1
	x3	LSL #2N	A-B	1
	x3	LSL #2N	A+B	0
	x0	LSL #2N+1	A+B	0
	x1	LSL #2N	A-B	1
	x2	LSL #2N	A+0	1
	x3			

High speed multipliers:-

where multiplication performance is very important, more hardware resource must be dedicated to it. some embedded system the ARM core is used to perform real time digital signal process to general control function.



Control structures:-



An instruction PLA. This unit uses some of the instruction bits to define ctrl an internal cycle counts to of operation.

ARM Instruction set:-

* Arm Processor support 6 data types
  - 8 bit signed & unsigned bytes
  - 16 bit signed & unsigned half words.

these are aligned on 2 byte
  - 32 bit signed & unsigned words

Arm instruction are all 32 bit words must be word aligned.

Thumb instruction are half words must align on 2 byte boundary

Multiply Instruction

ARM multiply instruction produce 32-bit binary number, hold of 2 -32 bit result of multiplying in register The result is a 64 bit product

Binary Encoding

Cond 0000 mul S Rd/RdHi Rn/RdLo Rs 1001 Rm

000 MUL Multiply (32 bit) Rd = (Rm * Rs)

001 MLA multiply accumulate (Rm * Rs)

100 VMULL unsigned multiply

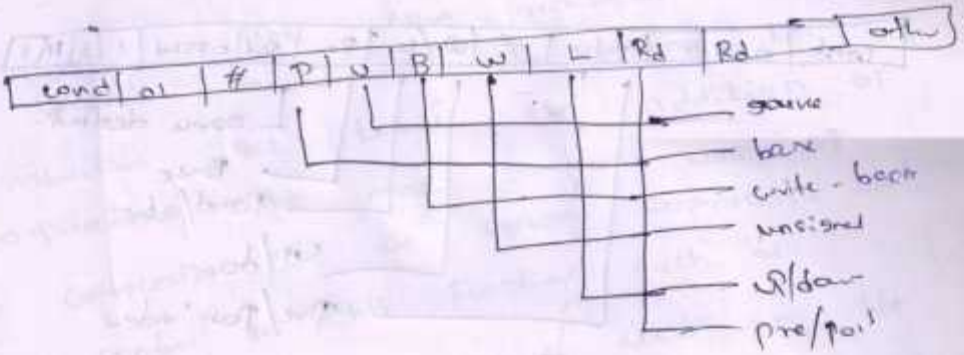
101 UMLAL unsigned multiply

110 SMULL signed multiply

Single word & unsigned byte data transfer instruction

These instructions are the most flexible way to transfer single byte or words of data b/w ARM register & memory.

Provided that register has been initialized to point some where near the reserved memory



The Pre index Form of instruction

LDSRISTR <cond> {B} Rd, Rn <offset>

LDRISTR <cond> {B} {T} Rd [Rn]

To store a byte in r0

LDR r1, UART_ADD

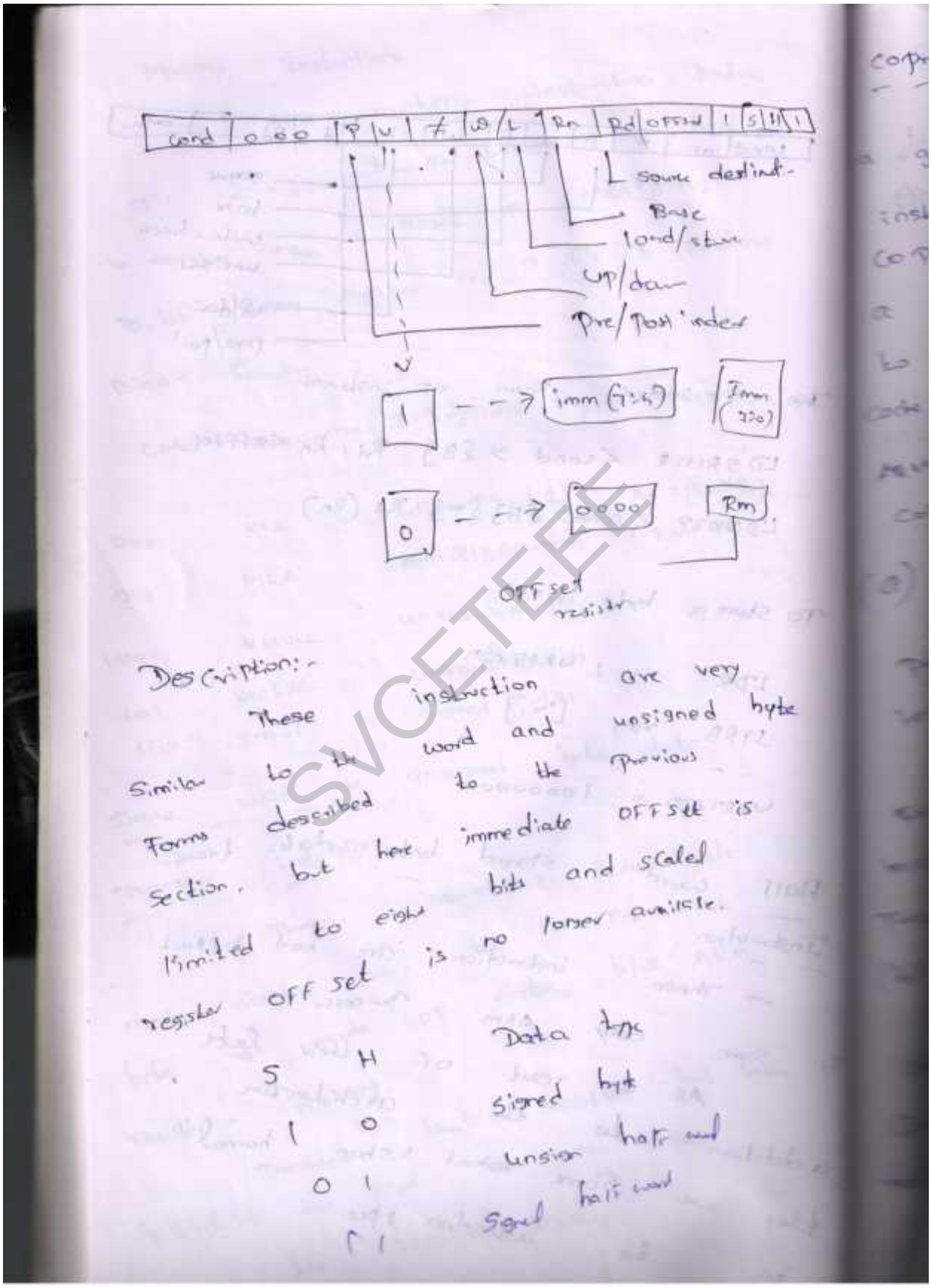
STRB r0, [r1]

UART_ADD EQU 1000000

Half word & signed byte data transfer Instruction

→ These instruction are not supported by some early ARM processor

As a result of their late addition to the architecture they are Some what clumsy horned to instruction spec in


cond
0 0 0
P
U
W
L
Rn
Rd
offset
1
S
H
1
Source destinat-
Base
load/store
up/down
Pre/Post index
I
imm (7:5)
Imm (2:0)
0
0000
Rm
OFFset register
Description:-
These instruction are very
Similar to the word and unsigned byte
forms described to the previous
section. but here immediate OFFSet is
limited to eight bits and scaled
register OFF set is no longer availble.
S H
1 0
0 1
1 1
Data type
signed byt
unsign half word
Signed half word

Coprocessor Instruction:-

The ARM architecture support a general mechanism for extend the instruction set through the addition of Coprocessor. The most common use of a coprocessor is the system coprocessor used to control to chip function such as cache & memory management unit on the ARM720 A floating point RAM ARM coprocessor has been develop

a) Coprocessor register:-

Arm coprocessor have their own private register sets & is controlled by instruction that mirror the instruction. The Arm has sole responsibility for controlling how go the coprocessor instruction are concerned with data transfer and data

(b) Coprocessor Data operation:-

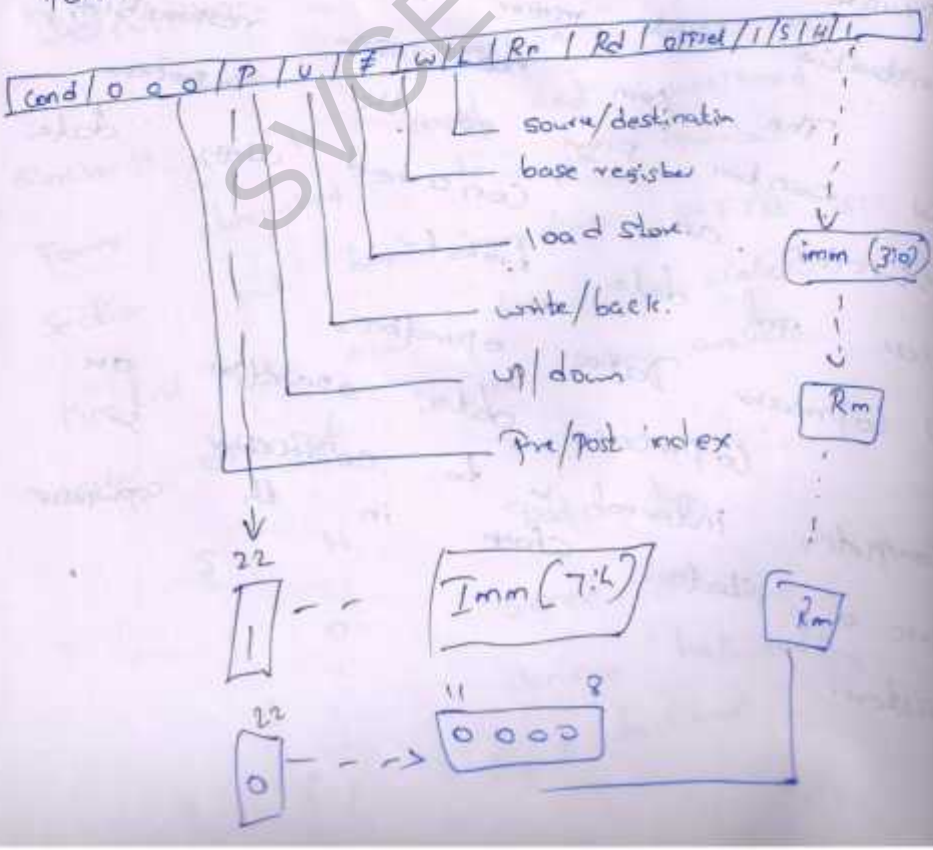
Coprocessor data operation are completely internal to coprocessor & cause a change in the coprocessor state

register.

Half word & signed byte data transfer

These instructions are not support by some of the early ARM Processor As a result of their late addition to the architecture they are some what shoe horned in to the instruction space as indicated by the split immediate field.

The addressing modes available with these instruction are a subset of the unsigned byte & word forms.



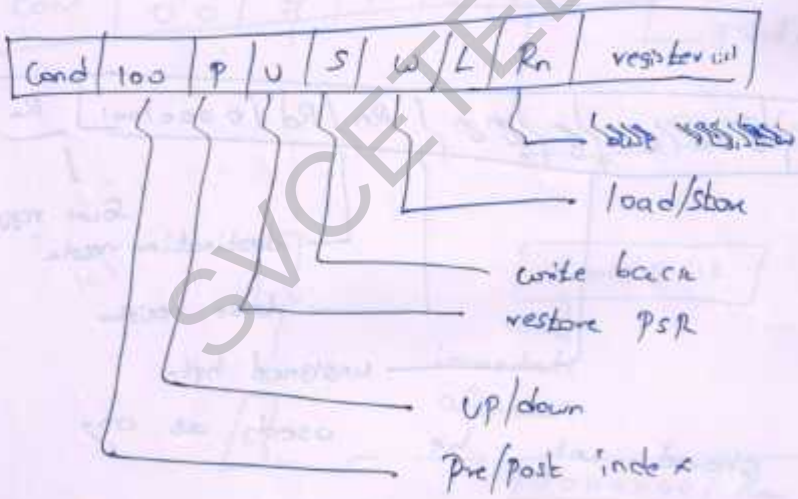
Description:-

These instruction are very similar to word & unsigned byte form described in previous section

S	H	Data type
1	0	signed byte
0	1	unsigned half word
1	1	signed half word.

Multiple Register Transfer instruction:-

ARM Multiple transfer instruction allow any subset

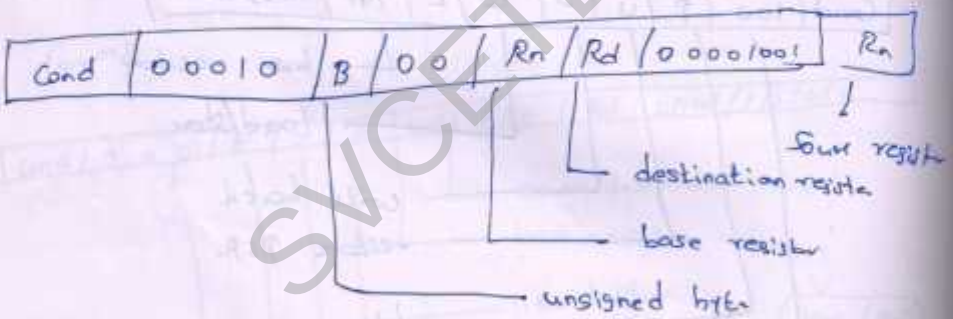


Description:-

The register list in bottom 16 bit of instruction include a bit for each visible register with bit 0 control whether or not to is transfer & so on bit is which control transfer of

Swap memory → Register Instruction.

Swap instruction combine a load of a word or an unsigned byte in a single instruction. Normally the two transfer are combined in to atomic memory operation cannot be split by an external memory access so there fore the instruction can be used as basis of Semaphore mechanism.



- pc should not be used as any of register
- Base register should not be same as either source or destination (Rd)
- when it is unnecessary to save or modify content of CPSR of current mode

General Register to status Register:

The operand which may be register or rotate 8 bit immediate as same way of immediate form of operand 2. in data processing instruction.

MSR {<cond>} CPSR_f ISPSR # <32 bit>

MSR {<cond>} CPSR_<Field> ISPSR <Field>

Unit V Completed

31/5/12