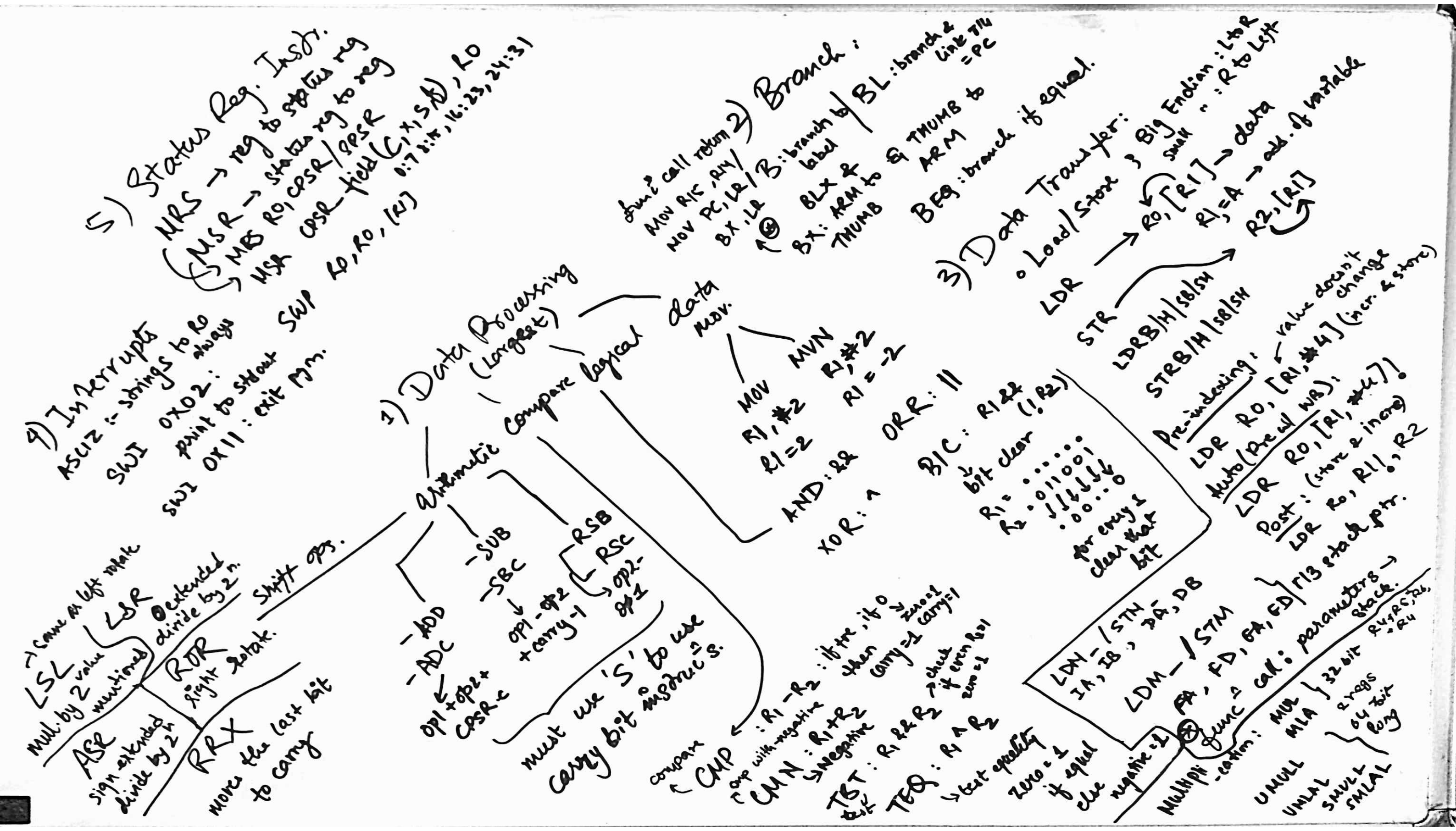
5) Status Reg. Instr.
MRS → reg to status reg
MSR → status reg to reg
MRS R0, CPSR/SPSR
MSR CPSR_field(C,X,S,F), R0
0:7 8:15, 16:23, 24:31
SWP R0, R0, [R1]
4) Interrupts
ASCIZ :- strings to R0 always
SWI 0X02: print to stdout
SWI 0X11: exit pgm.
1) Data Processing (Largest)
Arithmetic
Compare
logical
data mov.
MOV R1, #2 R1=2
MVN R1, #2 R1 = -2
AND: &&
ORR: ||
XOR: ^
BIC: R1 && (!R2)
bit clear
for every 1 clear that bit
-ADD
-ADC
op1+op2+CPSR-c
-SUB
-SBC
op1-op2+carry-1
-RSB
-RSC
op2-op1
must use 'S' to use carry bit instruc's.
Shift ops.
LSL mul. by 2^value mentioned
LSR extended divide by 2^n
same as left rotate
ASR sign extended divide by 2^n
ROR right rotate.
RRX moves the last bit to carry
compare
CMP: R1-R2 : if +ve, if 0 then carry=1 zero=1 carry=1
cmp with negative
CMN: R1+R2 negative
TST: R1 && R2 check if even R=0 zero=1
test
TEQ: R1 ^ R2 test equality zero=1 if equal else negative=1
2) Branch:
B: branch to label
BL: branch & link R14 = PC
func call return
MOV R15, R14/ MOV PC, LR/ BX, LR
BLX & BX: ARM to THUMB & THUMB to ARM
BEQ: branch if equal.
3) Data Transfer:
Load/Store
Big Endian: L to R
Small ": R to Left
LDR → R0, [R1] → data
R1 = A → add. of variable
STR → R2, [R1]
LDRB/H/SB/SH
STRB/H/SB/SH
Pre-indexing: LDR R0, [R1, #4] value doesn't change
Auto (Pre w/ WB): LDR R0, [R1, #4]! (incr. & store)
Post: (store & incre) LDR R0, R1!, R2
LDM_/STM_ IA, IB, DA, DB
LDM_/STM_ FA, FD, EA, ED
R13 stack ptr.
func call: parameters → stack. R4, R5, R6, R4
Multiplication: MUL MLA } 32 bit
UMULL UMLAL SMULL SMLAL } 2 regs 64 bit long

Locality of Reference:-

- Spatial: fetch nearby locations also when getting some data
- Temporal: set of instr. likely to be referenced SOON.

Cache Mapping:-

1) Direct Mapped:

2) Set Associative:

3) Fully associative:

Word = block size × word length

line = no. of lines in cache = cache size / line/block size

Set = line / set size (no. of sets in cache)

Tag = Main mem bits − word + set/line bits

Valid bit:-

- for each Block
- 0: initial value / data changed in main mem.
- 1: when data is loaded.

Write Hit:-

1) Write Through:
→ Both updated
→ consistency ✓
→ Latency ↑

2) Write Back:
→ Change CACHE only
→ Dirty bit (modified bit) = 1 (in cache)
→ while "replacing" check dirty bit & update in main mem. at that time.

Adv:
→ latency ↓
→ write everything at once

Cache Miss:-

→ data not in cache
→ miss penalty :/
→ Latency: time for 1st word retrieval
Bandwidth: retrieval of rest of the blocks.

Read Miss:-

1) Load through/early restart:
→ DON'T wait to load everything
→ continue once word is found & retrieval runs in //.

Write Miss:-

1) Write no allocate:
→ write directly to mem.
→ valid bit = 0
2) Write Allocate:
→ load data written to cache
→ easy access if same data needed again

Types of Misses:-

1) Compulsory miss:
→ initial miss.

2) Capacity miss:
→ not enough space to store

3) Conflict Miss: *
→ discarded block needs to be retrieved again

Replacement Algos:-

1) Random
2) LRU
3) FIFO
4) LFU (least freq. used)

4) Coherence miss:
→ due to flushing to maintain coherence.

Optimisations:

1) ↑ block size
2) ↑ cache size
3) ↑ associativity
4) Multilevel cache
5) Read priority > write
6) Avoid address translation when indexing cache.

1, 2, 3: ↓ miss rate
4, 5: ↓ miss penalty
6: ↓ Hit time

vishwa

Size ↑ speed ↓ capacity ↑
Cost ↓ Access time ↑

DATE

Performance Analysis :-

Avg. Mem. Access Time (AMAT) = Hit + Miss rate x miss penalty

↓

access + recieve + request

request time + $\frac{\text{Block size}}{\text{bus size}}$ + access time

(recieve)

CPU Execution Time:

= IC x Cycle time x Overall CPI

Overall CPI = Base CPI + CPU stall + Mem stall
↳ taken 0

Mem stall = % mem access x % miss rate x miss penalty

for unified cache : mem stall = (1 + % mem access) x % miss rate x miss penalty

for not unified cache :

mem stall = % access x % d-cache miss x penalty
+ % i-cache miss x penalty

for sep. mem access % :-

mem stall = % access x % load x % d cache miss x read penalty
+ % access x % store x % d-cache x write penalty
+
% i-cache miss x read penalty

PES UNIVERSITY