OP_CODE

Function	ALU_OP	Output Y
AND	000	A AND B
OR	001	A OR B
XOR	010	A XOR B
NOT	011	!A
ADD	100	A + B
SUB	101	A - B
A	110	A
B	111	B

Unsigned range: 0 to $2^n - 1$

Signed Range: $-\ 2^{n-1}$ to $2^{n-1} - 1$

Sign [0 is positive]
Exponent [exponent + 127 in binary]
Mantissa [drop leading one, pad/cut to 23]

Set: AND, Clear: OR, Invert: XOR
Big Endian: First first, Little Endian: Last first

FETCH: Instruction is fetched
REG: Register values are loaded
ALU: Operation is executed
MEM: Memory stage, Branch sets PC
WRITE: Write back to registers

Branch hazard: lines that aren't supposed to executed are flushed after PC is set to branch
Data hazard: stalling occurs to wait for data resolved in previous line to be written back

Cache slot: $(int)(slot\_num/4)\ \%\ N$

Timing | Efficiency: ($T_{eff} = cache_{access} + miss_{rate} * mem_{access}$) | Clock Frequency $= 1/D$, $D = signal\ delay$

a	Pipeline cycle time	Max delay among stages + inter-register delay
b	Non-pipeline cycle time	Sum of all delays among stages
c	Speed up ratio	*b / a*
d	Pipeline time for X tasks	$a + (X - 1) * a$
e	Sequential time for X tasks	$X * b$
f	Latency	Time for a single cycle: *d* || *b*
g	Throughput for X instructions	X/f

ALU_OP: What operation is performed?
Imm_nReg: Is ALU_B an immediate? (0/1)
RdA_addr: Address of A (rN/XX)
RdB_addr: Address of B (rN/XX)
DM_RD: Memory read (1/0)
DM_WR: Memory write (1/0)
MEM_nALU: Writing back an ALU result (1), data_mem output (XX), or neither (0)?
Wr: Are we writing back to a register? (0/1)
Wr_addr: Writing to what register? (rN/XX)
PC: Program counter (+4/branch_name)

CMP: SUB, TST: AND

Stack Push: STR rN,[r13],#-4
Stack Pop: LDR rN,[r13,#4]!

Spatial and Temporal Locality: Data has spatial locality when it's close to other frequently accessed data in memory. The cache takes advantage of this as it will load related data inherently. Data has temporal locality when it's accessed repeatedly in a short period of time.

Misc: RAM is not part of the CPU, Moore's Law: 2 years, overflow in addition is detected on carry-out from MSB, N bits of binary can represent 2^N numbers, computers *do not* know whether a number is signed or unsigned, N bit decoders have 2^N outputs, in sequential digital circuits outputs depend on the previous, full adders have carry-in while half adders do not, the program counter determines the address of the next instruction, PC is set to *branch_name* when BAL *branch_name* is executed, CPUs do not have a connection to the main memory, fully associative cache mapping has a cache register per memory block.

ARM Commands — Appending S to command not listed as [F] listed updates flags

Cmd	Operation	Affects	Description
ADD	Add	R	op1 + op2
SUB	Subtract	R	op1 - op2
ADC	Add with carry	R	op1 + op2 + carry flag
SBC	Subtract with carry	R	op1 - op2, and if carry is clear, -1 (C = 0 for borrow)
CMP	Compare	F	op1 - op2: updates flags, doesn’t store result
MOV	Move	R	Loads op1 into the destination
MVN	Move not	R	Moves not(operand 1) into the destination register
AND	And	R	op1 AND op2
ORR	Or	R	op1 OR op2
EOR	Exclusive or	R	op1 XOR op2
TST	Test	F	Performs an AND on operands 1 and 2
LDR	LoaD Register	R	LDR r1,=0xFEED_1973
LDR	&	M	LDR r1,[r2],#4 loads mem at [r2] into r1, adds 4 to r2
LSR	Logical shift R	F	Regular shift right
ASR	Arithmetic shift R	F	Shifts and replaces sign bit with original sign bit
ROR	Rotation R	F	Shifts right with rotation
LSL	Logical shift L	F	Regular shift left
BAL	Branch always	BMI	If minus
BEQ	If equal	BPL	If plus
BNE	Not equal	BVS	If overflow set
BGT	Greater than	BVC	If no overflow
BGE	Greater or equal		
BLE	Less than or equal		
BLT	Less than		