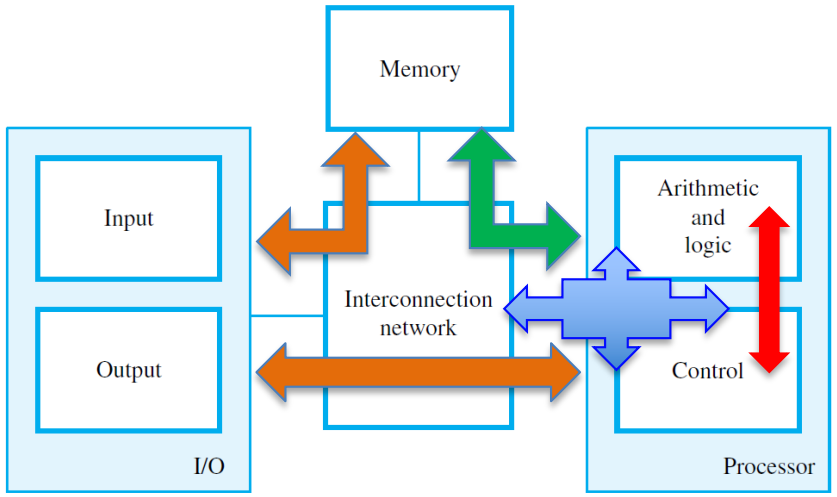
2.3 Instructions and Sequencing

- Four instructions categories in most processors:
  1. **Data transfer: memory to/from processor**
  2. **Input/output transfer: I/O to/from processor/memory**
  3. **Arithmetic and logic operations**
  4. **Program sequencing and control**



36 ARM Thumb Instructions (TH3-4)

ADC	Add with Carry
ADD	Add
AND	AND
ASR	Arithmetic Shift Right
B	Unconditional branch
B*xx*	Conditional branch
BIC	Bit Clear
BL	Branch and Link
BX	Branch and Exchange
CMN	Compare Negative
CMP	Compare
EOR	EOR

LDMIA	Load multiple
LDR	Load word
LDRB	Load byte
LDRH	Load halfword
LSL	Logical Shift Left
LDSB	Load sign-extended byte
LDSH	Load sign-extended halfword
LSR	Logical Shift Right
MOV	Move register
MUL	Multiply
MVN	Move Negative register

NEG	Negate
ORR	OR
POP	Pop registers
PUSH	Push registers
ROR	Rotate Right
SBC	Subtract with Carry
STMIA	Store Multiple
STR	Store word
STRB	Store byte
STRH	Store halfword
SWI	Software Interrupt
SUB	Subtract
TST	Test bits

2.3.4 Load/Store Architecture

- Two operations to access (read or write) memory
  - **Load** : register ← memory
  - **Store** : memory ← register
  - Source: no change
  - Destination: overwritten
  - Same operations to access input and output
  - A characteristic of the Reduced Instruction Set Computer (RISC)

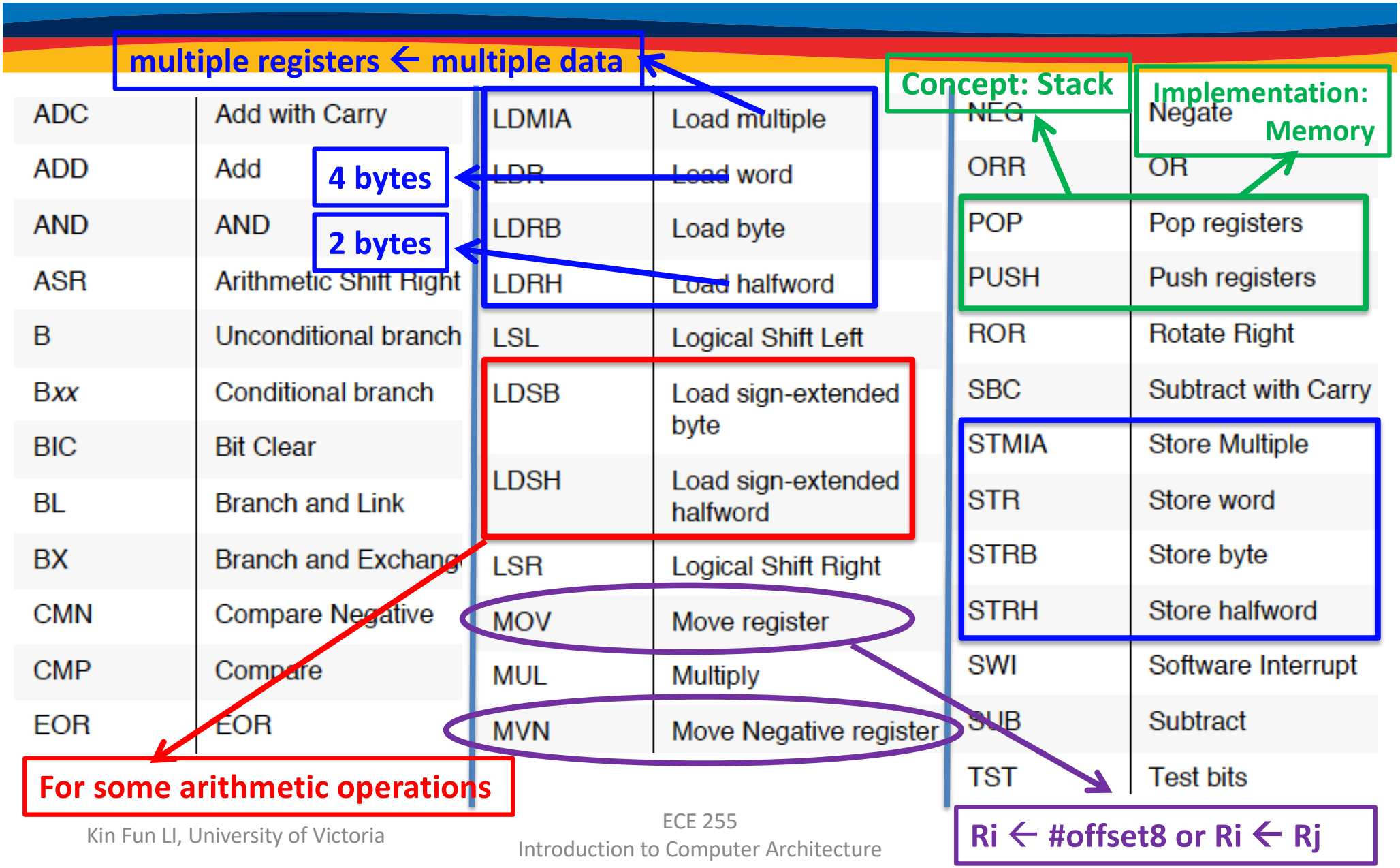
Load/Store Architecture

- ALU ← operands
  - only from #imm and registers; not from memory
- A characteristic of RISC vs Complex Instruction Set Computer (CISC can use memory as ALU operands)

Storing data operands in registers: advantages ?

- Faster access to operands than from memory: no bus delay
- Temporary results do not need to write back to memory
- Free up system bus for other uses (in parallel with ALU operation)

1,2: Thumb Memory/Input-Output Access Instructions



multiple Load/Store (TH5-15)

- LDMIA Rb!, { Rlist }
  - Load the registers specified by Rlist, starting at the base address in Rb. Write back the new base address.

- STMIA R0!, {R3-R7}
  - Store the contents of registers R3-R7 starting at the address specified in R0, incrementing the addresses for each word. Write back the updated value of R0.

load/store with register offset (TH5-7)

- LDR Rd, [Rb, Ro] / LDRB Rd, [Rb, Ro]
  - Pre-indexed word load: Calculate the source address by adding together the value in Rb and the value in Ro. Load the contents / byte value of the address into Rd.

- STR R3, [R2, R6]
  - Store word in R3 at the address formed by adding R6 to R2.

load/store with immediate offset (TH5-9)

- LDR Rd, [Rb, #Imm] / LDRB Rd, [Rb, #Imm]
  - Calculate the source address by adding together the value in Rb and Imm. Load word / byte value into Rd from the address.

- STRB R1, [R0, #13]
  - Store the lower 8 bits of R1 at the address formed by adding 13 to R0.

3: Thumb Data Processing Instructions (TH3-4)

ADC	Add with Carry
ADD	Add
AND	AND
ASR	Arithmetic Shift Right
B	Unconditional branch
B*xx*	Conditional branch
BIC	Bit Clear
BL	Branch and Link
BX	Branch and Exchange
CMN	Compare Negative
CMP	Compare
EOR	EOR

LDMIA	Load multiple
LDR	Load word
LDRB	Load byte
LDRH	Load halfword
LSL	Logical Shift Left
LDSB	Load sign-extended byte
LDSH	Load sign-extended halfword
LSR	Logical Shift Right
MOV	Move register
MUL	Multiply
MVN	Move Negative register

NEG	Negate
ORR	OR
POP	Pop registers
PUSH	Push registers
ROR	Rotate Right
SBC	Subtract with Carry
STMIA	Store Multiple
STR	Store word
STRB	Store byte
STRH	Store halfword
SWI	Software Interrupt
SUB	Subtract
TST	Test bits

Logic in Green

Arithmetic in Red

Bit Operation in Blue

ADD/SUB (TH5-2)

- ADD Rd, Rs, Rn
  - Add contents of Rn to contents of Rs. Place result in Rd.
- ADD Rd, Rs, #Offset3
  - Add 3-bit immediate value to contents of Rs. Place result in Rd.

ADD/SUB/CMP/MOV Immediate (TH5-3)

- MOV Rd, #Offset8
  - Move 8-bit immediate value into Rd.
- CMP Rd, #Offset8
  - Compare contents of Rd with 8-bit immediate value.
- SUB Rd, #Offset8
  - Subtract 8-bit immediate value from contents of Rd and place the result in Rd

ALU operations: AND/LSL/CMP (TH5-4)

- AND Rd, Rs
  - Rd:= Rd AND Rs
- LSL Rd, Rs
  - Rd := Rd << Rs
- CMP Rd, Rs
  - Set condition codes on Rd - Rs

4: Thumb Branch/Control Instructions (TH3-4)

ADC	Add with Carry
ADD	Add
AND	AND
ASR	Arithmetic Shift Right
B	Unconditional branch
B*xx*	Conditional branch
BIC	Bit Clear
BL	Branch and Link
BX	Branch and Exchange
CMN	Compare Negative
CMP	Compare
EOR	EOR

Programming

Subroutine Call

LDMIA	Load multiple
LDR	Load word
LDRB	Load byte
LDRH	Load halfword
LSL	Logical Shift Left
LDSB	Load sign-extended byte
LDSH	Load sign-extended halfword
LSR	Logical Shift Right
MOV	Move register
MUL	Multiply
MVN	Move Negative register

NEG	Negate
ORR	OR
POP	Pop registers
PUSH	Push registers
ROR	Rotate Right
SBC	Subtract with Carry
STMIA	Store Multiple
STR	Store word
STRB	Store byte
STRH	Store halfword
SWI	Software Interrupt
SUB	Subtract
TST	Test bits

Operating System

Conditional Branch (TH5-16)

Cond	THUMB assembler	ARM equivalent	Action
0000	BEQ label	BEQ label	Branch if Z set (equal)
0001	BNE label	BNE label	Branch if Z clear (not equal)
0010	BCS label	BCS label	Branch if C set (unsigned higher or same)
0011	BCC label	BCC label	Branch if C clear (unsigned lower)
0100	BMI label	BMI label	Branch if N set (negative)
0101	BPL label	BPL label	Branch if N clear (positive or zero)
0110	BVS label	BVS label	Branch if V set (overflow)
0111	BVC label	BVC label	Branch if V clear (no overflow)
1000	BHI label	BHI label	Branch if C set and Z clear (unsigned higher)
1001	BLS label	BLS label	Branch if C clear or Z set (unsigned lower or same)
1010	BGE label	BGE label	Branch if N set and V set, or N clear and V clear (greater or equal)
1011	BLT label	BLT label	Branch if N set and V clear, or N clear and V set (less than)
1100	BGT label	BGT label	Branch if Z clear, and either N set and V set or N clear and V clear (greater than)
1101	BLE label	BLE label	Branch if Z set, or N set and V clear, or N clear and V set (less than or equal)

Branch and Link (TH5-19)

- BL label
  - Unconditionally Branch to label and place following instruction address in R14, the Link Register.