ARM data instructions

- Basic format:

  `ADD R0,R1,R2`

  – Computes R1+R2, stores in R0.

- Immediate operand:

  `ADD R0,R1,#2`

  – Computes R1+ 2, stores in R0.

ARM data instructions

- ADD, ADC : add (w. carry)
- SUB, SBC : subtract (w. carry)
- MUL: multiply
- AND, ORR, EOR
- BIC : bit clear
- LSL, LSR : logical shift left/right
- ROR : rotate right

ARM load/store instructions

- LDR, LDRH, LDRB : load (half-word, byte)
- STR, STRH, STRB : store (half-word, byte)
- Addressing modes:
  - register indirect : `LDR R0,[R1]`
  - with constant : `LDR R0,[R1,#4]`

ARM LDR instruction

- Load from memory into a register

```
LDR R8,[R10]
```

Example: C assignments

- C:

```
x = (a + b) - c;
```

- Assembler:

```
LDR R4,=A          ; get address for a
LDR R0,[R4]              ; get value of a
LDR R4,=B          ; get address for b, reusing r4
LDR R1,[R4]              ; get value of b
ADD R3,R0,R1             ; compute a+b
LDR R4,=C          ; get address for c
LDR R2,[R4]              ; get value of c
```

C assignment, cont’d.

```
SUB R3,R3,R2      ; complete computation of x
LDR R4,=X         ; get address for x
STR R3,[R4]       ; store value of x
```

Example: C assignment

- C:

```
y = a*(b+c);
```

- Assembler:

```
LDR R4,=B ; get address for b
LDR R0,[R4] ; get value of b
LDR R4,=C ; get address for c
LDR R1,[R4] ; get value of c
ADD R2,R0,R1 ; compute partial result
LDR R4,=A ; get address for a
LDR R0,[R4] ; get value of a
```

C assignment, cont’d.

```
MUL R2,R2,R0 ; compute final value for y
LDR R4,=Y ; get address for y
STR R2,[R4] ; store y
```

Example: C assignment

- C:

```
Z = (A << 2) |  (B & 15);
```

- Assembler:

```
LDR R4,=A ; get address for a
LDR R0,[R4] ; get value of a
LSL R5,R0,#2 ; perform shift
LDR R4,=B ; get address for b
LDR R1,[R4] ; get value of b
AND R1,R1,#15 ; perform AND
ORR R1,R5,R1 ; perform OR
```

C assignment, cont’d.

```
LDR R4,=Z ; get address for z
STR R1,[R4] ; store value for z
```

Example: if statement

- C:

```
if (a > b) { x = 5; y = c + d; } else x = c - d;
```

- Assembler:

```
; compute and test condition
   LDR R4,=A ; get address for a
   LDR R0,[R4] ; get value of a
   LDR R4,=B ; get address for b
   LDR R1,[R4] ; get value for b
   CMP R0,R1 ;
   BLE fblock ;
```