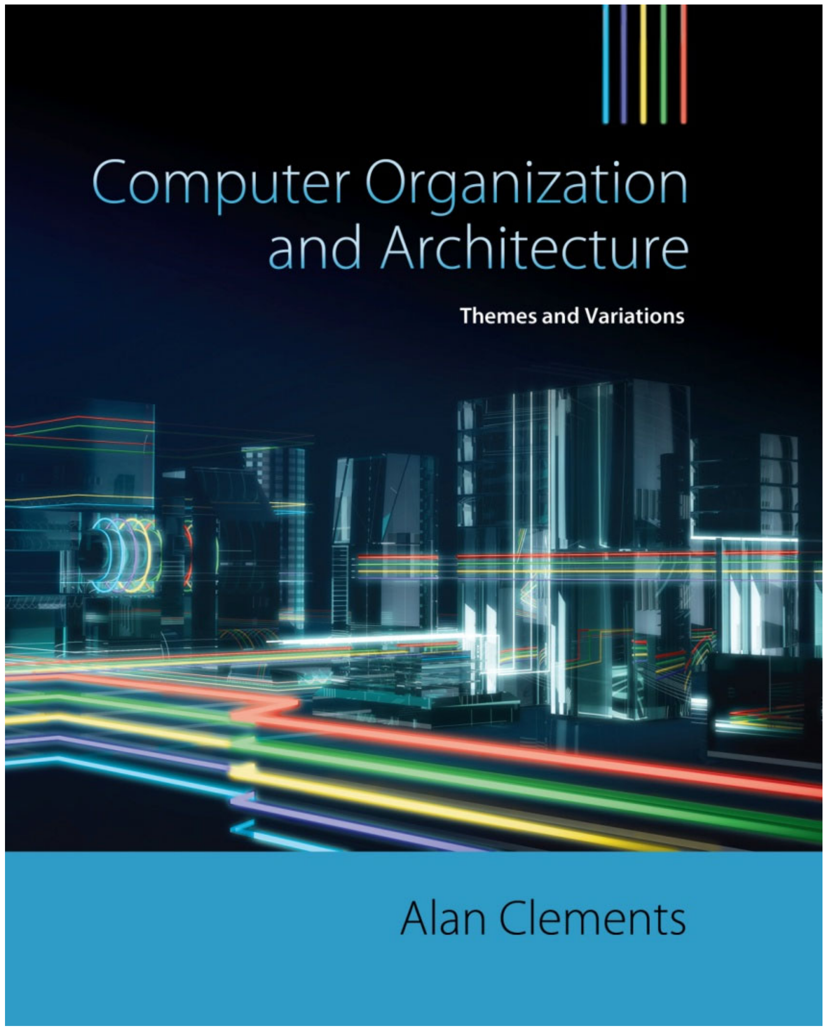
Part D

Chapter 3

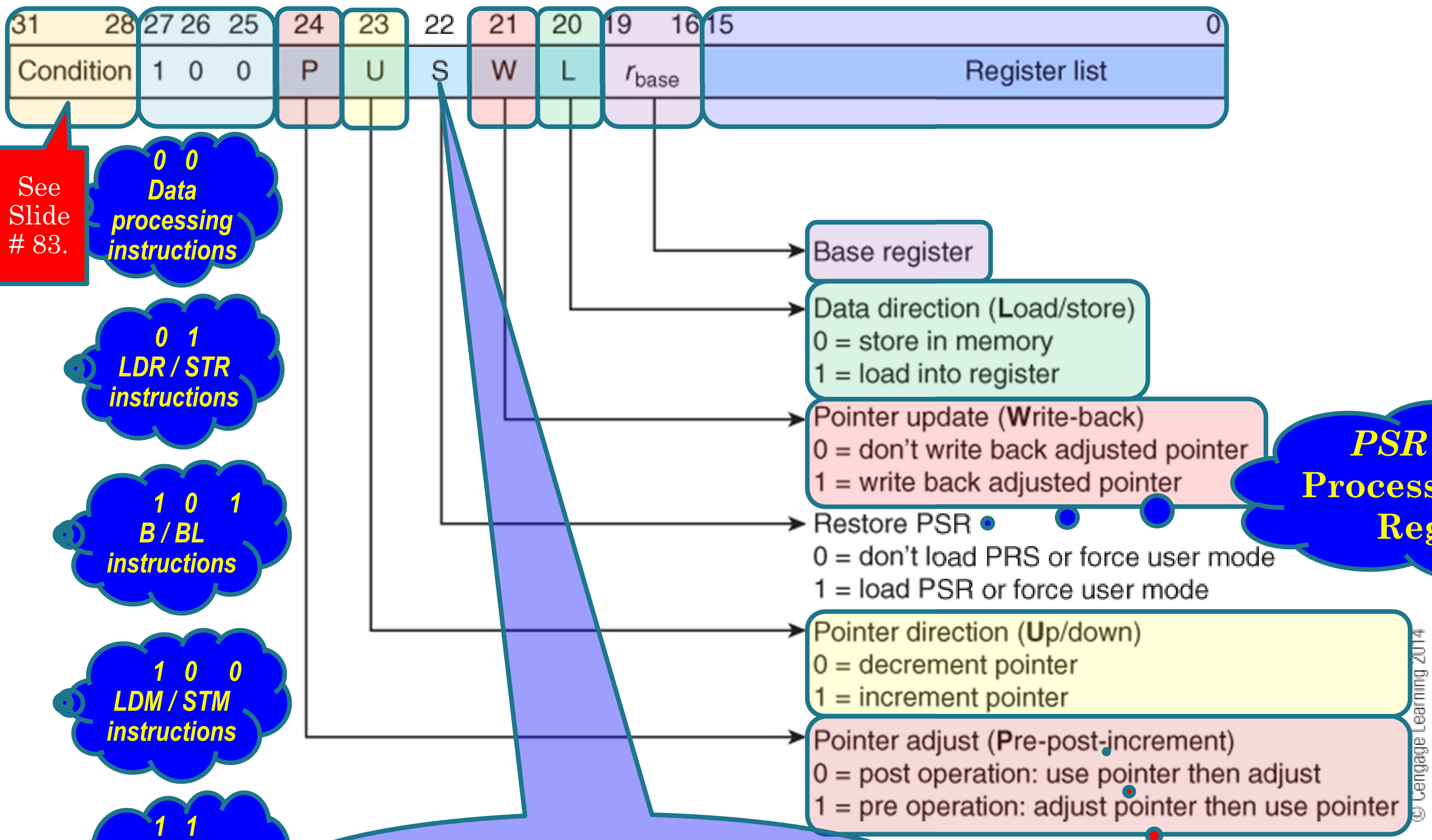
Architecture and Organization



CENGAGE Learning

Block Move Instructions Encoding/Decoding

FIGURE 3.58 Encoding ARM's block move instructions


31 28 27 26 25 24 23 22 21 20 19 16 15 0
Condition
1 0 0
P
U
S
W
L
r_{base}
Register list
Base register
Data direction (Load/store)
0 = store in memory
1 = load into register
Pointer update (Write-back)
0 = don't write back adjusted pointer
1 = write back adjusted pointer
Restore PSR
0 = don't load PRS or force user mode
1 = load PSR or force user mode
Pointer direction (Up/down)
0 = decrement pointer
1 = increment pointer
Pointer adjust (Pre-post-increment)
0 = post operation: use pointer then adjust
1 = pre operation: adjust pointer then use pointer
© Cengage Learning 2014

See Slide # 83.

0 0 Data processing instructions

0 1 LDR / STR instructions

1 0 1 B / BL instructions

1 0 0 LDM / STM instructions

1 1 Coprocessor and SW interrupt instructions

PSR means Processor Status Register

During this course, it will always be 0

Will not be covered

Should be "Pre-post-update"

Block Move Instructions Encoding Example

ARM Instruction: **STMFD r13!,{r0-r4,r10}**

Condition = 1110 (always – unconditional)

P = 1 (D**B**: adjust pointer then use pointer)

U = 0 (**D**B: decrement)

S = 0 (user mode)

W = 1 (write-back adjusted pointer)

L = 0 (store)

r_{base} = 1101 (r13)

Register list (r15, r14, …., r2, r1, r0) = 0000 0100 0001 1111

1110 1001 0010 1101 0000 0100 0001 1111

0xE92D041F

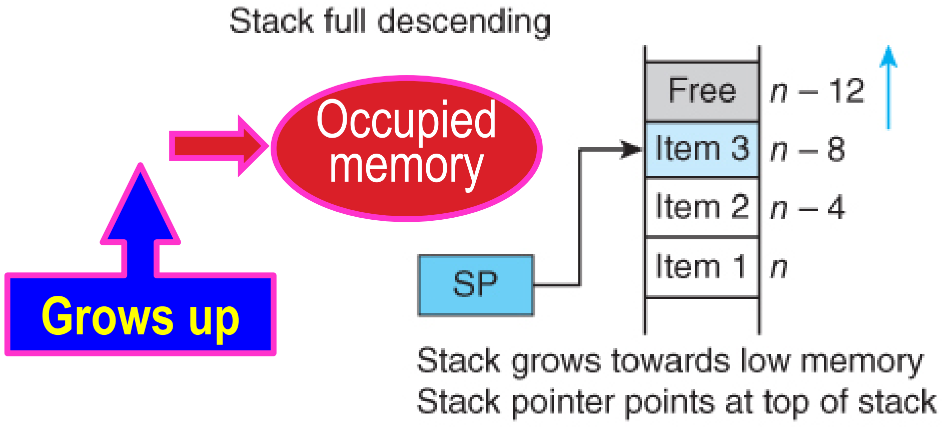
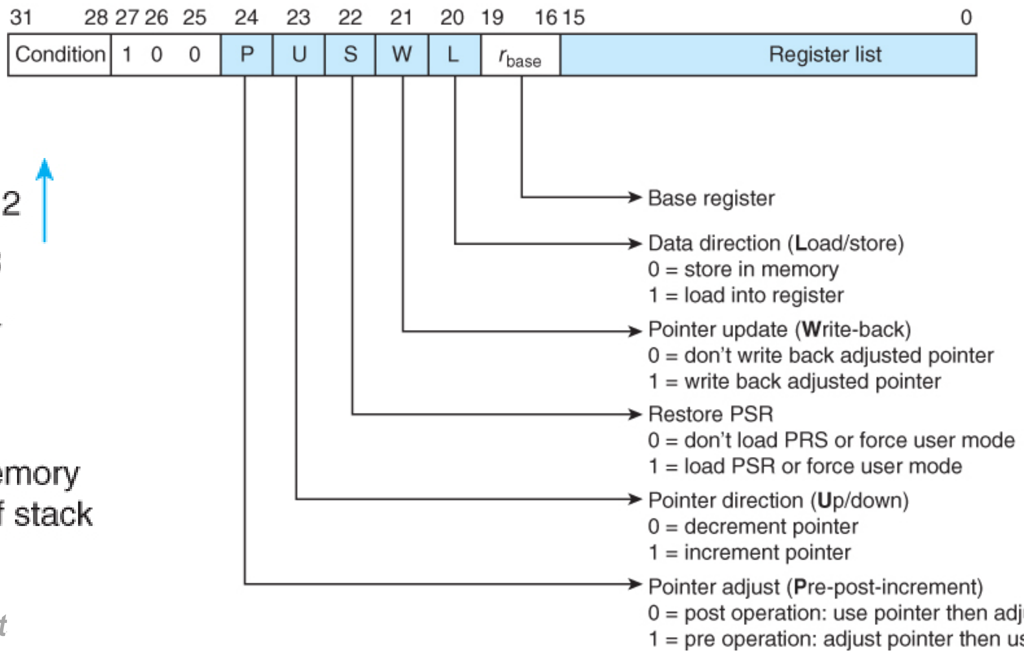


FIGURE 3.58 Encoding ARM's block move instructions


31
28 27 26 25
24
23
22
21
20 19
16 15
0
Condition
1 0 0
P
U
S
W
L
r_base
Register list
Base register
Data direction (Load/store)
0 = store in memory
1 = load into register
Pointer update (Write-back)
0 = don't write back adjusted pointer
1 = write back adjusted pointer
Restore PSR
0 = don't load PRS or force user mode
1 = load PSR or force user mode
Pointer direction (Up/down)
0 = decrement pointer
1 = increment pointer
Pointer adjust (Pre-post-increment)
0 = post operation: use pointer then adjust
1 = pre operation: adjust pointer then use pointer

Block Move Instructions Encoding Example

ARM Instruction: `LDMFD r13!,{r0-r4,r10}`

```
Condition = 1110 (always – unconditional)
    P = 0 (IA: use pointer then adjust)
    U = 1 (IA: increment)
    S = 0 (user mode)
    W = 1 (write-back adjusted pointer)
    L = 1 (load)
```

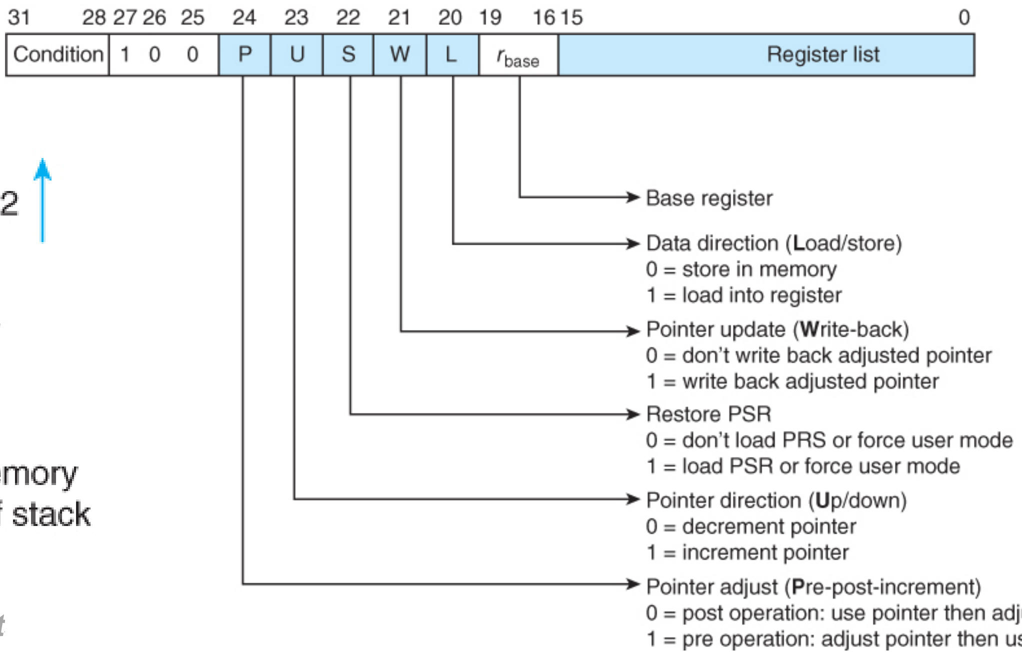
r_{base} = 1101 (r13)

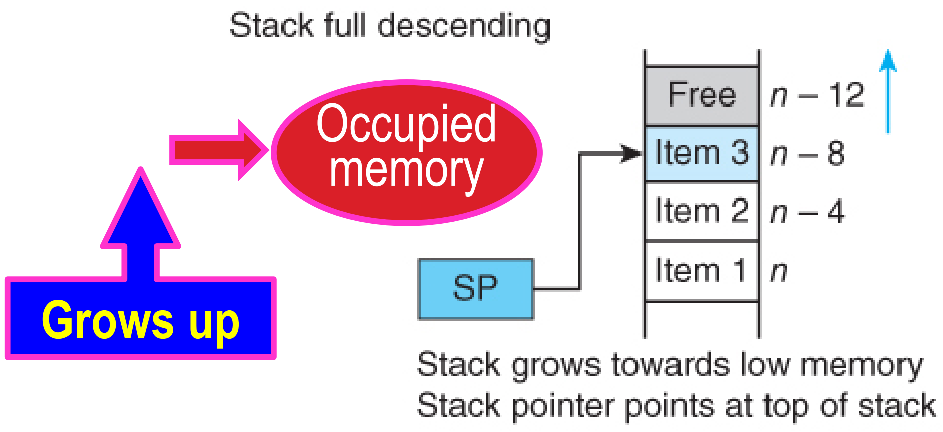
Register list (r15, r14, …., r2, r1, r0) = 0000 0100 0001 1111

```
1110 1000 1011 1101 0000 0100 0001 1111
```

0xE8BD041F

FIGURE 3.58 Encoding ARM's block move instructions


31
28 27 26 25
24
23
22
21
20 19
16 15
0
Condition
1 0 0
P
U
S
W
L
r_{base}
Register list
Base register
Data direction (Load/store)
0 = store in memory
1 = load into register
Pointer update (Write-back)
0 = don't write back adjusted pointer
1 = write back adjusted pointer
Restore PSR
0 = don't load PRS or force user mode
1 = load PSR or force user mode
Pointer direction (Up/down)
0 = decrement pointer
1 = increment pointer
Pointer adjust (Pre-post-increment)
0 = post operation: use pointer then adjust
1 = pre operation: adjust pointer then use pointer



*This slide is a modified version of the original author's slide (**A. Clement***

Block Move Instructions Decoding Example

Decode the ARM machine language **0x08855555**

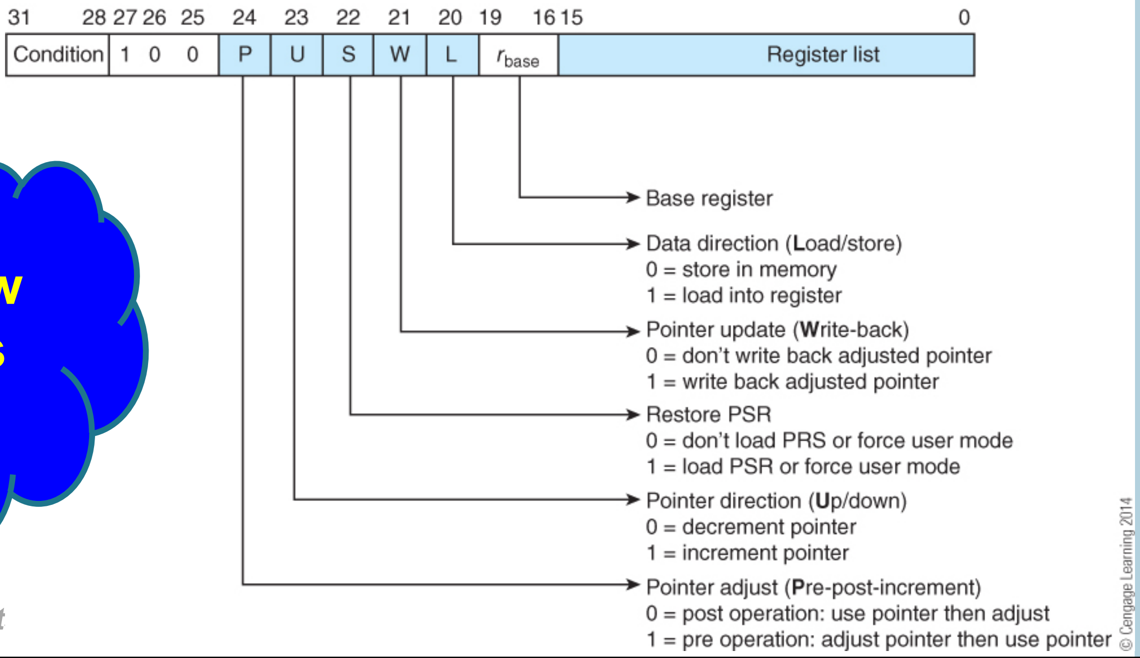
```
    0000 1000 1000 0101 0101 0101 0101 0101
Condition = 0000 (EQ)
    P = 0 (IA: use pointer then adjust)
    U = 1 (IA: increment)
    S = 0 (user mode)
    W = 0 (do not write-back adjusted pointer)
    L = 0 (store)
    rbase = 0101 (r5)
    Register list (r15, r14, …., r2, r1, r0) = 0101 0101 0101 0101
```

ARM Instruction: `STMEQIA r5,{r0,r2,r4,r6,r8,r10,r12,r14}`

It can also be
STMIAEQ
STMEQEA
STMEAEQ

Review slides 177

FIGURE 3.58 Encoding ARM's block move instructions


31
28 27 26 25
24
23
22
21
20 19
16 15
0
Condition
1 0 0
P
U
S
W
L
rbase
Register list
Base register
Data direction (Load/store)
0 = store in memory
1 = load into register
Pointer update (Write-back)
0 = don't write back adjusted pointer
1 = write back adjusted pointer
Restore PSR
0 = don't load PRS or force user mode
1 = load PSR or force user mode
Pointer direction (Up/down)
0 = decrement pointer
1 = increment pointer
Pointer adjust (Pre-post-increment)
0 = post operation: use pointer then adjust
1 = pre operation: adjust pointer then use pointer
© Cengage Learning 2014

*This slide is a modified version of the original author's slide (**A. Clement***

Block Move Instructions Decoding Example

Decode the ARM machine language **0x99922222**

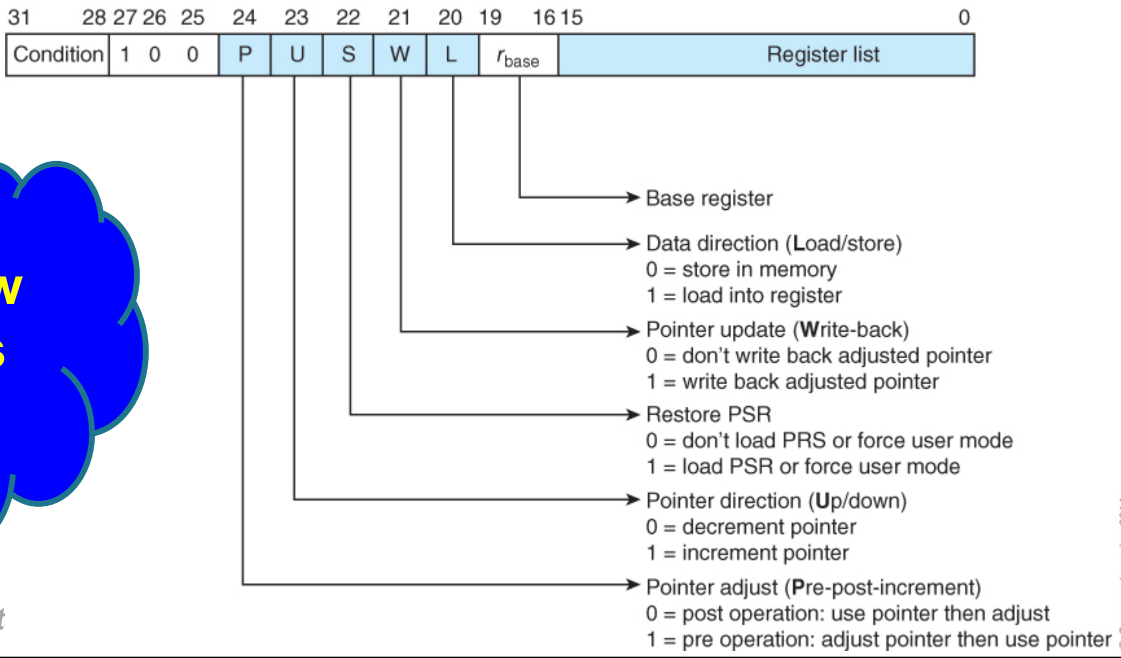
```
    1001 1001 1001 0010 0010 0010 0010 0010
Condition = 1001 (LS)
    P = 1 (IB: adjust pointer then use pointer)
    U = 1 (IB: increment)
    S = 0 (user mode)
    W = 0 (do not write-back adjusted pointer)
    L = 1 (load)
    r_base = 0010 (r2)
    Register list (r15, r14, …., r2, r1, r0) = 0010 0010 0010 0010
```

ARM Instruction: `LDMLSIB` **`r2`**`,{r1,r5,r9,r13}`

It can also be
LDMIBLS
LDMLSED
LDMEDLS

Review slides 176

FIGURE 3.58 Encoding ARM's block move instructions



31 28 27 26 25 24 23 22 21 20 19 16 15 0
Condition | 1 0 0 | P | U | S | W | L | r_base | Register list
Base register
Data direction (Load/store)
0 = store in memory
1 = load into register
Pointer update (Write-back)
0 = don't write back adjusted pointer
1 = write back adjusted pointer
Restore PSR
0 = don't load PRS or force user mode
1 = load PSR or force user mode
Pointer direction (Up/down)
0 = decrement pointer
1 = increment pointer
Pointer adjust (Pre-post-increment)
0 = post operation: use pointer then adjust
1 = pre operation: adjust pointer then use pointer
© Cengage Learning 2014

*This slide is a modified version of the original author's slide (**A. Clement***