COL216
Computer Architecture

ARM Assembly Programming continued

13th Jan, 2022

Question about ARM instructions

- Range of constants in various instructions?
- Accessing bits/bytes in a register?
- Other instructions between cmp and branch?
- Working with stack?
- Machine language (encoding of assembly language)?

Instruction format

- An instruction has many parts
- These are called fields
- Arrangement of fields is called instruction format

Format for DP instructions

Rd <= Rn + operand2

add r1, r2, r3

add r1, r2, #3

Format for DP instructions

Rd <= Rn + operand2

add r1, r2, r3

add r1, r2, #3

F = 00 for DP instructions

Shift operations on registers

Shift type:

- LSL logical shift left
- LSR logical shift right
- ASR arithmetic shift right
- ROR rotate right

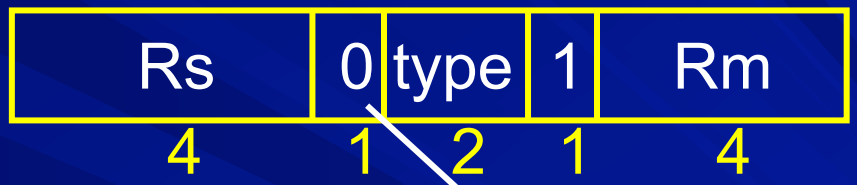
Shift amount

- 5 bit unsigned constant # 0 . . # 31
- 4 bit register number r0 . . r15

Operand2 : Register with shift

Rm, LSL #4

Rm, LSL Rs

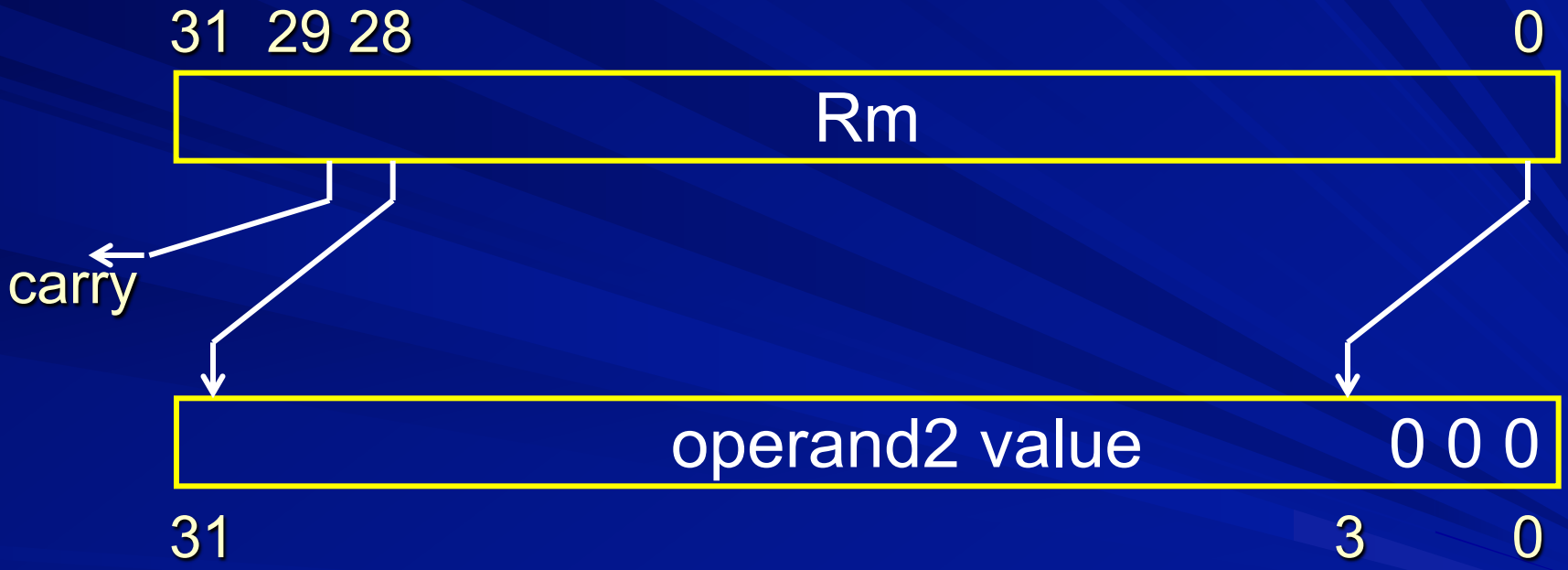


Shift type:

- 00 LSL logical shift left
- 01 LSR logical shift right
- 10 ASR arithmetic shift right
- 11 ROR rotate right

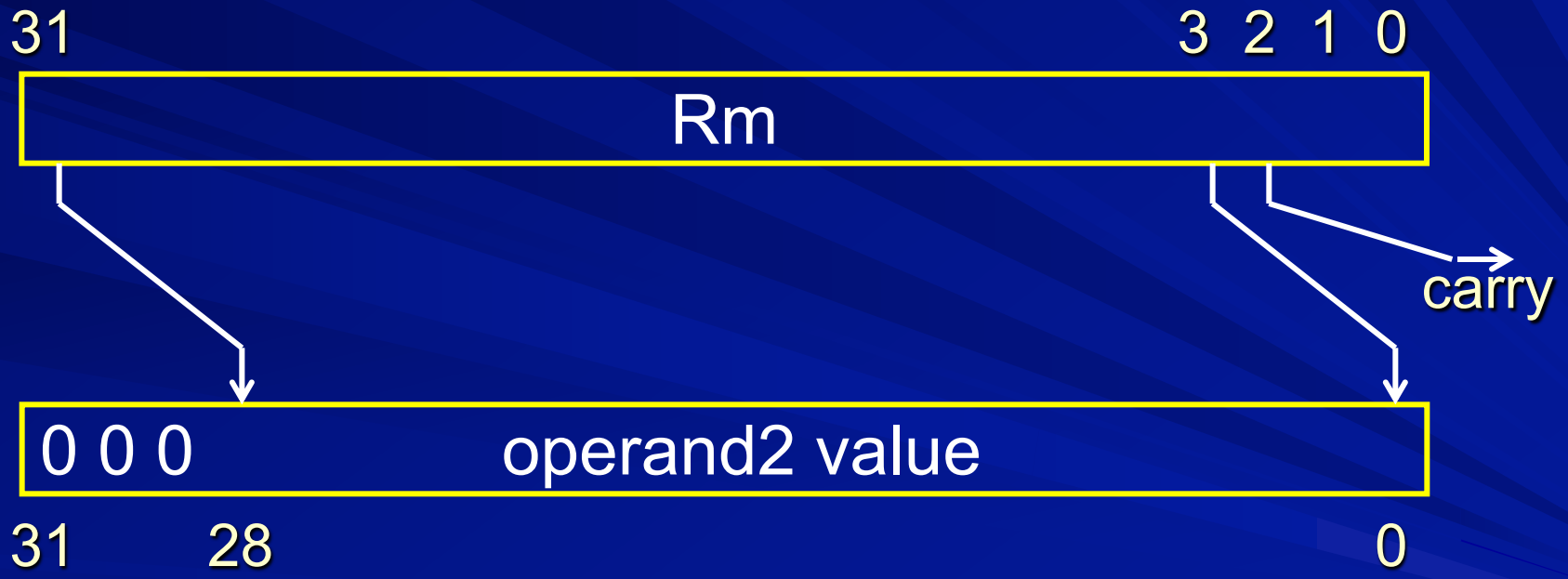
Shift types: Logical Shift Left

- LSL # 3



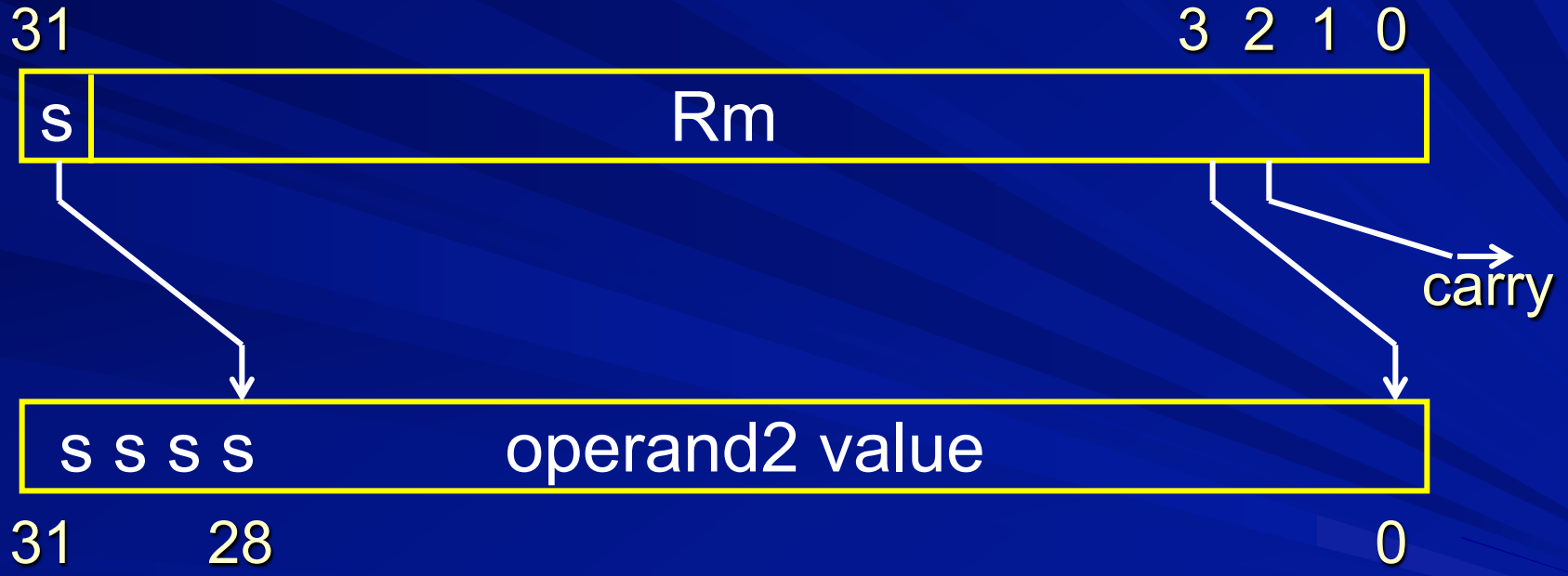
Shift types: Logical Shift Right

- LSR # 3



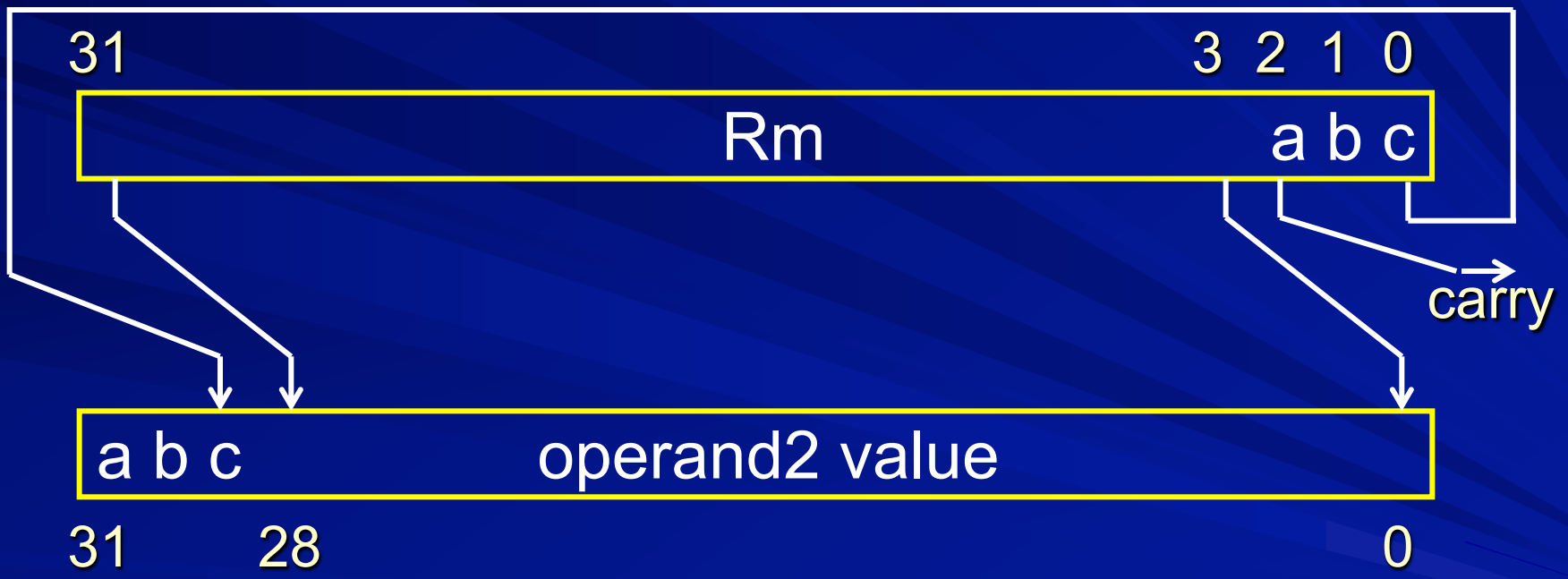
Shift types: Arithmetic Shift Right

- ASR # 3



Shift types: Rotate Right

- ROR # 3



Rotate operation on constant

- rot is a 4 bit unsigned constant (0 to 15), specifying how much to rotate
- Imm is a 8 bit constant (0 to 255) which is zero extended to 32 bits and rotated right by 2*rot bits (that is, by 0 to 30 bits)

e.g., operand2 = #400 => Imm = 100, rot = 15

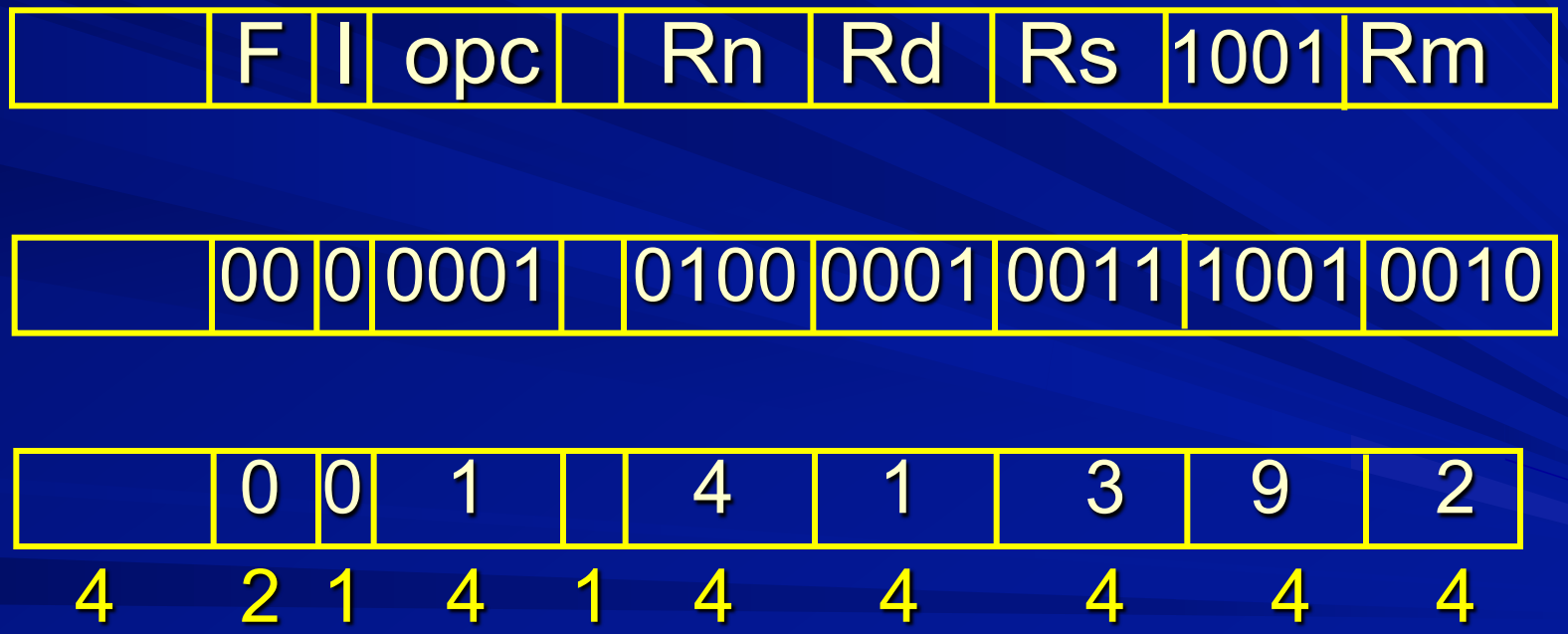
operand2 = #800 => Imm = 50, rot = ?

Format for “mul”

mul r1, r2, r3 r1 <= r2 * r3

Multiply accumulate

mla r1, r2, r3, r4 r1 <= r2 * r3 + r4



Multiplying by a constant

mul r1, r2, # 10

➔

mov r3, # 10

mul r1, r2, r3

Multiplying by a power of 2

mul r1, r2, # 16

➔

mov r1, r2, LSL # 4

LSL = Logical Shift Left

Format for DT instructions

Rd <= Memory [Rn + offset]

F = 01

12 bit offset field in DT instructions

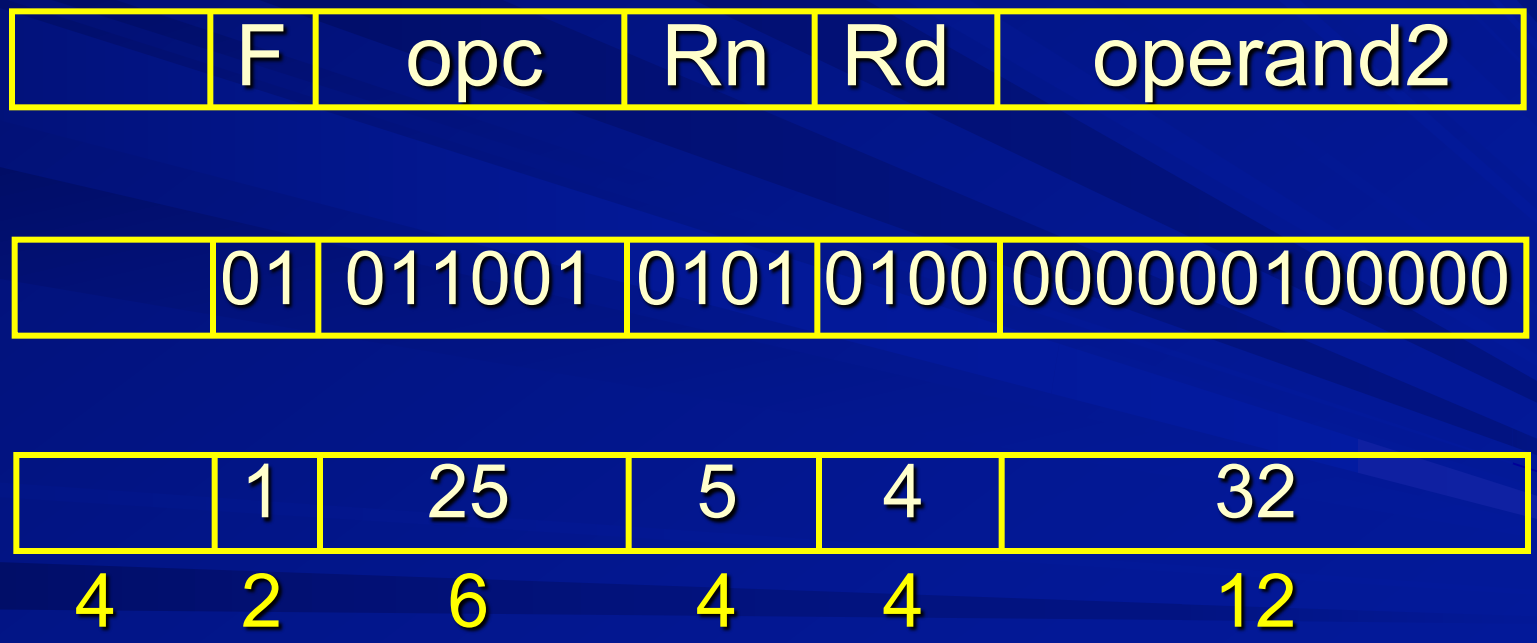
- 12 bit unsigned constant

or

- 4 bit register number, 8 bit shift specification (same as constant shift spec of DP instructions)

Example of DT instruction

ldr r4, [r5, #32]



Indexing an array

```
mul  r4, r5, #4
add  r2, r4, r2
ldr   r6, [r2, #0]
```

➔

```
add r2, r2, r5, LSL #2
ldr   r6, [r2]
```

➔

```
ldr   r6, [r2, r5, LSL #2]
```

Instruction similar to ldr

- str (opcode = 24)

rd is the source, not destination

memory address is the destination, not source

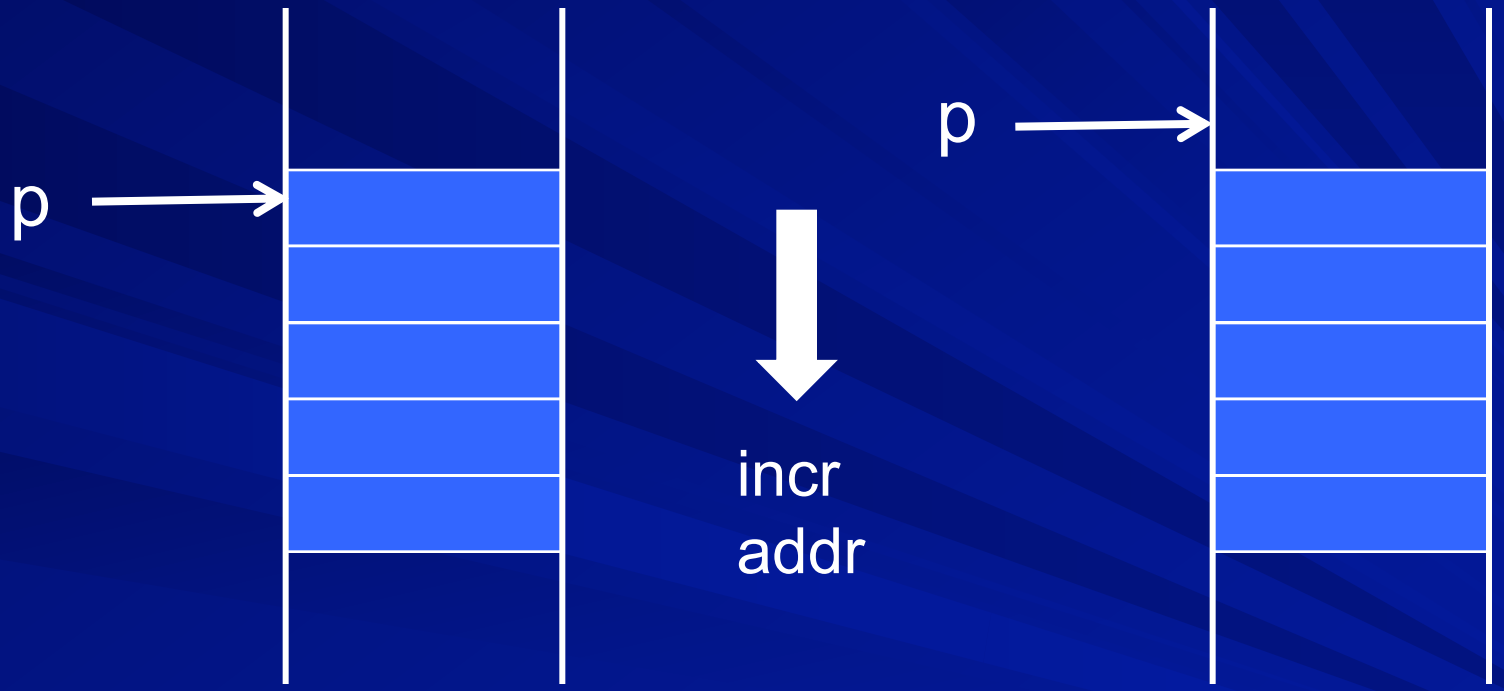
Opcode field in DT instructions

- 6 opcode bits specify I, P, U, B, W, L
  - I (immediate): constant or register with shift
  - P (pre/post) pre or post indexing
  - U (up/down) whether to add or subtract offset
  - B (byte) byte or word transfer
  - W (write back) whether to write back address into base register (Rn) or not
  - L (load/store) load from memory or store into memory

DT instruction examples

```
ldr  r4, [r5, - r6]
str  r4, [r5, r6, LSL # 2]
ldrb  r4, [r5, # 32] !
strb  r4, [r5, # -32]
ldr  r4, [r5], r6
str  r4, [r5], r6, LSL # 2
```

Using auto-increment/decrement



get/pop:	post increment	pre increment
put/push:	pre decrement	post decrement

Thank you