Week 12

Block Move → allows you to move consecutive memory to a group of registers; replaces having to LDR into a register and then increment the pointer in separate instructions

LDM → block move from memory to registers (similar to LDR)
STM → block move from registers to memory (similar to STR)

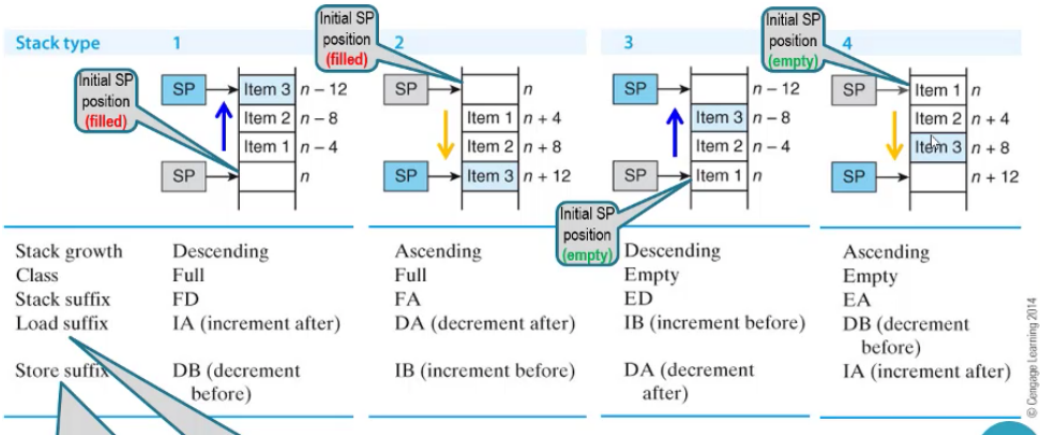
Operations frequently combined with Stack operations.

Syntax:

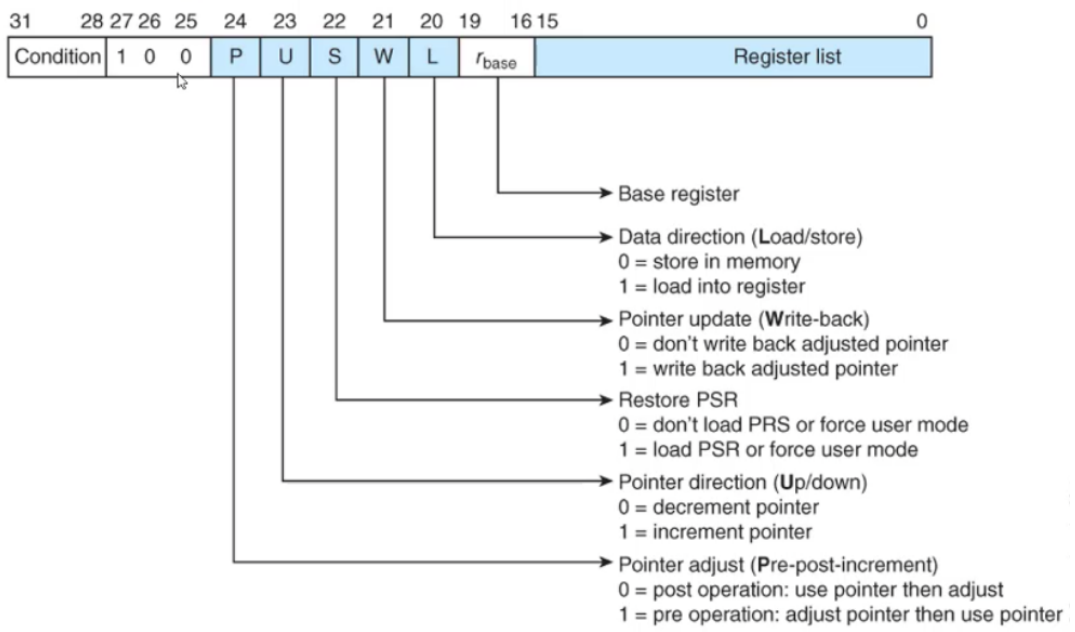
```
STMIA r0!,{r1-r3,r5}
```

- Register list is put in curly brackets; can have a range
- IA = indicates r0 is incremented after (IA) the transfer
- Order doesn't matter - ARM stores lowest numbered register first at lowest memory address, and so on

Block Move and Stack Operations:



Encoding Block Move Instruction



Subroutine → like a function

- Set of instructions that may be repeatedly called
- Processors saves address of next instruction and loads program counter with first instruction in subroutine
- At end of subroutine, a return from subroutine instruction causes the processors to return to the point immediately following the subroutine call

BL (branch with link) automatically saves the return address in register r14.

- This branch instruction has a 24-bit signed program counter relative offset
- EXAM: review branch instruction!!!

You can use a stack to push the return address onto the stack and branching to the target address. (typically used by CISC processor).

- Once subroutine code is completed, you pop the return address from the stack and copy it to the PC

Note: STR and STR MUL has a pipelining effect of +12 (adding 12 to PC) unlike other instructions that are +8.

Leaf Routine → doesn't call another routine, it's at the end of the tree

- Return address saved in link register r14
- Return to calling point is made with a MOV pc, lr

If we want to save previous values that are stored in registers before a subroutine, we push the values stored in the registers onto the stack and then pop the values and load them into the same registers after.

- We can use LDR multiple and STR multiple (block moves) to save register values once entering a subroutine and restore registers before returning subroutine