Embedded Systems Design
Lecture 7

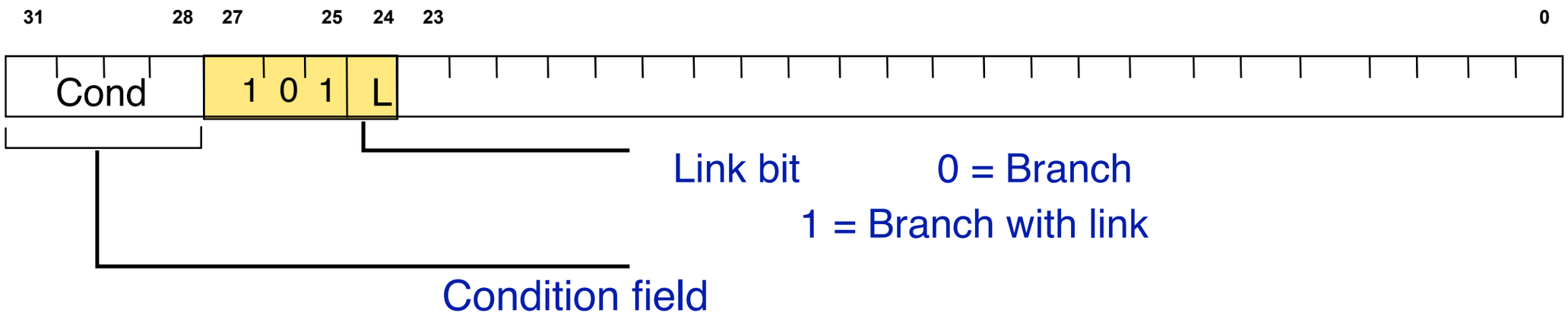
Yongsoo Joo

KESL
Kookmin Univ. Embedded System Lab

Branch Instructions

Instruction	**B, BL**	**Branch, Branch Link**
Format	**B{L}{cond} <expression>** **{L}** is for **branch with link => PC is stored in R14(LR)** **{cond}** is the condition flag for conditional execution **<expression>** is the **destination** (offset)	
Behavior	**if {cond}** is met**, jumps to the destination calculated by <expression>, and PC value is stored into LR if {L} option** is used.	
Flag	Not affected	
Example	**B there** ; **unconditional jump** **BL sub+label** ; destination is calculated by sub+label **CMP R1,#0** ; **comparison between R1 and #0** **BEQ fred** ; **if R1 is #0, jump to fred**	

Branch Instructions



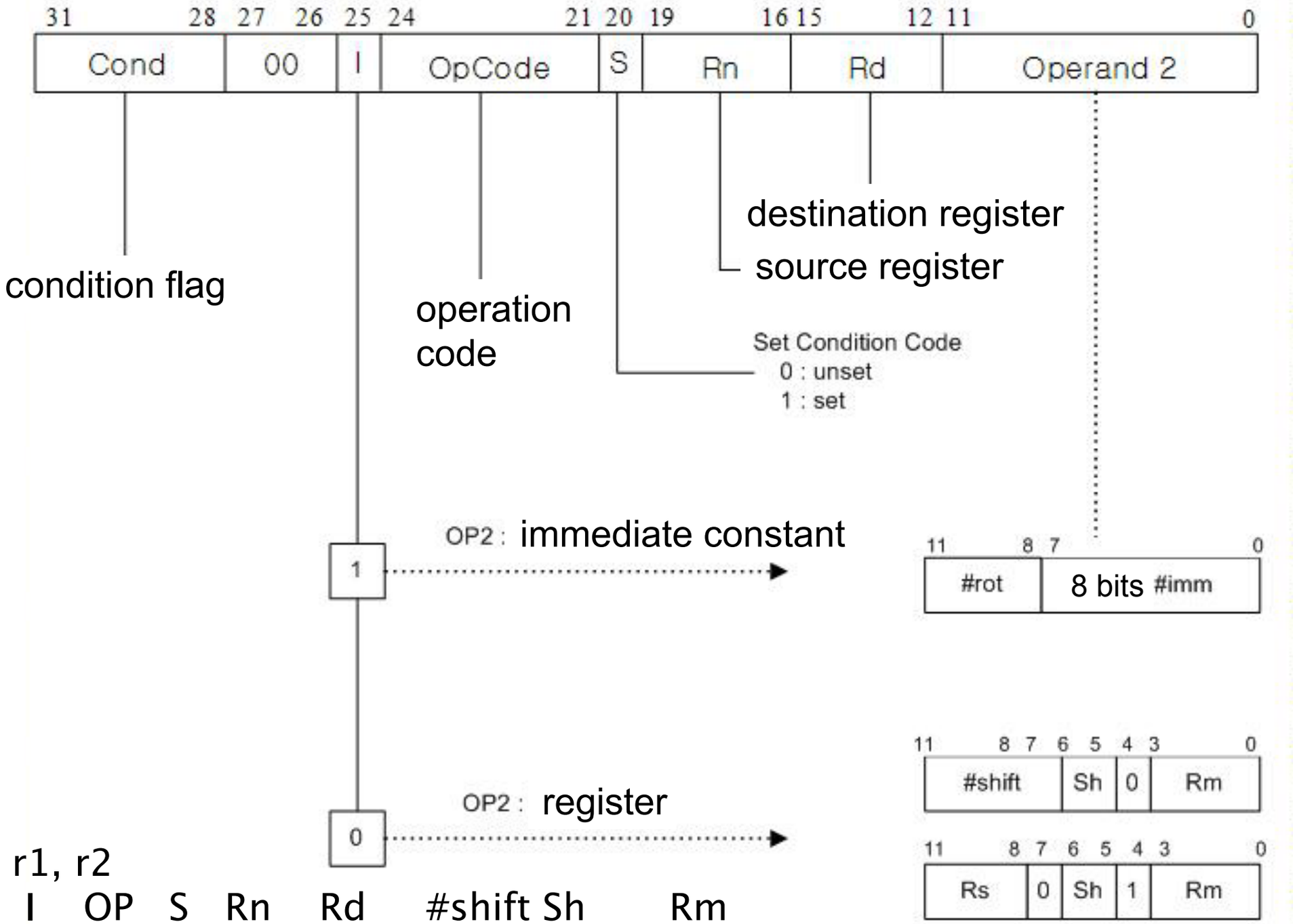
- Destination address calculation
  - Bits [23:0] are used as offset using PC as the basis
  - Offset calculation
    - MSB is the sign bit (+ or -)
    - The remaining 23 bits are used as offset with 2 bits left shift operation

Subroutines

- Branch and Link (BL) instruction is used
  - Stores the next address into LR
  - LR <= PC - 4 (Why PC - 4?)
- Returning from subroutine
  - *MOV pc, lr ; pc = lr*

			PC	
base + 0	BL there	IF	base+0	
base + 4	add R0, R1, R2	ID	base+4	
base + 8	STR R0, [R4]	EX	base+8 :	LR <= base + 4 = (PC = base+8) – 4)

Data Processing Instructions



ADD r0, r1, r2

COND I OP S Rn Rd #shift Sh Rm

1110 00 0 0010 0 0001 0000 00000 00 0 0010

AL

e0810002

Arithmetic Operations

- Arithmetic operation lists

opcode	inst	behavios	explanation
0100	ADD	Rd := Rn + Op2	Add
0101	ADC	Rd := Rn + Op2 + Carry	Add with Carry
0010	SUB	Rd := Rn – Op2	Subtract
0110	SBC	Rd := Rn – Op2 – 1 + Carry	Subtract with Carry
0011	RSB	Rd := Op2 - Rn	Reverse subtract
0111	RSC	Rd := Op2 – Rn – 1 + Carry	Reverse Subtract with Carry

- Examples
  - ADD r0, r1, r2
  - SUBGT r3, r3, #1
  - RSBLES r4, r5, #5

Logical Operations

- Logical operation lists

opcode	inst	behavior	explanation
0000	AND	Rd := Rn AND Op2	AND
1111	ORR	Rd := Rn OR Op2	OR
0001	EOR	Rd := Rn XOR Op2	Exclusive OR
1110	BIC	Rd := Rn AND (NOT Op2)	Bit Clear

- Examples
  - AND r0, r1, r2
  - BICEQ r2, r3, #7
  - EORS r1,r3,r0

Comparison Operations

- NO ‘S’ postfix to affect condition flags
- Comparison operation lists

opcode	inst	behavior	expanation
1010	CMN	CPSR flags := Rn + Op2	Compare Negative
1011	CMP	CPSR flags := Rn-Op2	Compare
1000	TEQ	CPSR flags := Rn EOR Op2	Test bitwise equality
1001	TST	CPSR flags := Rn AND Op2	Test bits

- Examples
  - CMP r0, r1
  - TSTEQ r2, #5

Data Move Operations

- Data move operation lists

opcode	inst	behavior	explanation
1101	MOV	Rd := Op2	Move register or constant
1111	MVN	Rd := 0xFFFFFFFF EOR Op2	Move Negative register

- Examples
  - MOV r0, r1 ; r1 -> r0
  - MOVS r2, #10 ; 10 -> r2
  - MVNEQ r1,#0 ; if zero flag set then 0 -> r1

Arithmetic Operation with Shift

- Benefit
  - Arithmetic operation on the operands where shift operation to 2nd operand is applied in a single instruction
  - Barrel shifter is used
- Arithmetic operation with shift format
  - Immediate value based

```
ADD  r5,  r5,  r3  LSL #3
```

  - Register value is used for shift bits

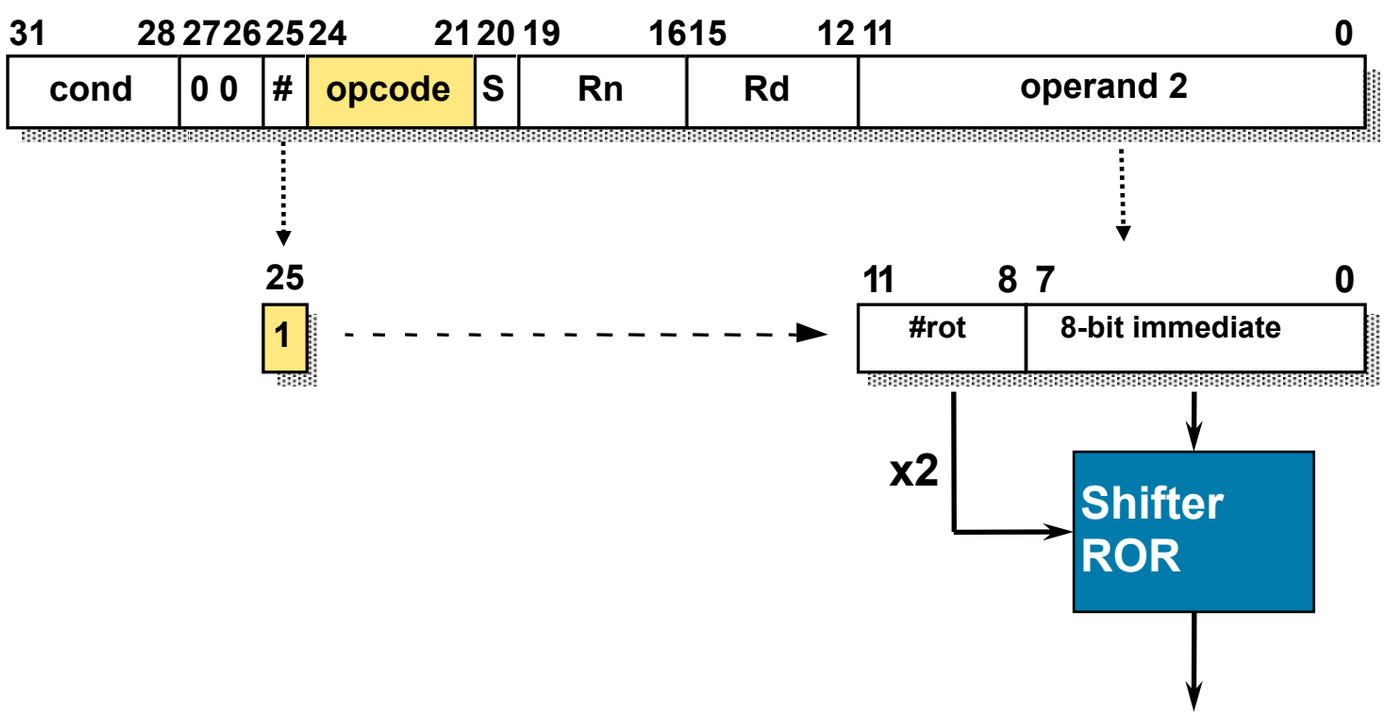
```
ADD  r5,  r5,  r3  LSL r2
```

Arithmetic Operation with Shift

- ASR: the signed value of the contents of a register divided by a power of two. It copies the sign bit into vacated bit positions on the left.
- LSL(=ASL): the value of a register is multiplied by a power of two.
- LSR: the unsigned value of a register is divided by a variable power of two.
  - Both instructions (LSL, LSR) insert zeros into the vacated bit positions.
- ROR: the value of the contents of a register rotated by a value.
  - The bits that are rotated off the right end are inserted into the vacated bit positions on the left.

Operand 2 and Immediate Value

- How we can use big immediate values in the limited space inside 32 bits instruction format?
  - A tricky method to represent immediate values using just 12 bits



2 x #rot bits of ROR is applied to the 8 bits immediate value

8-bit immediate = 0xFF
#rot = 0x4

Immediate value = 0xFF000000

Operand 2 and Immediate Value

- Valid immediate values

```
0x000000FF
0x00000FF0
0xFF000000
0xF000000F
```

- Invalid immediate values

```
0x000001FE
0xF000F000
0x55550000
```

PSR Transfer Instruction

- PSR (Program Status Register) needs special treatment
  - The only possible way to access is through moving register contents between general purpose registers and PSRs
  - MRS: Move PSR to Register
  - MSR: Move Register to PSR
- CPSR (Current Program Status Register)
  - CPSR can be accessed at all the operation modes
  - Only condition fields can be modified at USER mode
- SPSR (Saved Program Status Register)
  - Each operation mode has a dedicated SPSR
    - User mode does not have SPSR

PSR Transfer Instruction (MRS)

Inst	MRS	Move PSR to Registers
Format	MRS{cond} Rd, <PSR> {cond} is the condition for execution <PSR> is either CPSR or SPSR	
Behavior	Move PSR to Rd	
Example	MRS r0, CPSR ; r0 := CPSR	

PSR Transfer Instruction (MSR)

Inst	MSR	Move Register to PSR
Format	MSR{cond} <PSR[_fields]>, Rm {cond} is the condition for the execution <PSR> is either CPSR or SPSR [_fields] is specific fields in PSR (condition flag(f), control bits(c), ….)	
Behavior	Move Rm to <PSR[_fields]>	
Example	MSR CPSR, r0 ; CPSR := r0 MSR CPSR_f, r0 ; CPSR condition flags bits := r0 MSR CPSR_fc, r1 ; CPSR flags, control bits := r1 MSR CPSR_c, #0x80 ; CPSR control bits := 0x80	