Task 08- Step B

- Decode the following object code into assembly code:

```
00:     e3a00002
04:     eb000006
08:     e0811080
0c:     e2514005
10:     ca000000
14:     e0244004
18:     e3a00000
1c:     e3a07001
20:     ef000000
24:     e2000003
28:     e1a0f00e
```

Task 08- Step B

- Decode the following object code into assembly code:

```
00:      e3a00002
04:      eb000006
08:      e0811080
0c:      e2514005
10:      ca000000
14:      e0244004
18:      e3a00000
1c:      e3a07001
20:      ef000000
24:      e2000003
28:      e1a0f00e
```

```
1110 0011 1010 0000 0000 0000 0000 0010
```

Task 08- Step B

- Decode the following object code into assembly code:

```
00:      e3a00002

04:      eb000006

08:      e0811080

0c:      e2514005

10:      ca000000

14:      e0244004

18:      e3a00000

1c:      e3a07001

20:      ef000000

24:      e2000003

28:      e1a0f00e
```

```
1110 00 1 1101 0 0000 0000 0000 00000010
```

Task 08- Step B

- Decode the following object code into assembly code:

```
00:      e3a00002

04:      eb000006

08:      e0811080

0c:      e2514005

10:      ca000000

14:      e0244004

18:      e3a00000

1c:      e3a07001

20:      ef000000

24:      e2000003

28:      e1a0f00e
```

```
1110 00 1 1101 0 0000 0000 0000 00000010
 AL     I  MOV S  Rn   Rd  #rot   #2
```

Task 08- Step B

- Decode the following object code into assembly code:

```
00:      e3a00002

04:      eb000006

08:      e0811080

0c:      e2514005

10:      ca000000

14:      e0244004

18:      e3a00000

1c:      e3a07001

20:      ef000000

24:      e2000003

28:      e1a0f00e
```

```
1110 00 1 1101 0 0000 0000 0000 00000010
 AL     I  MOV S  Rn   Rd  #rot   #2      MOV r0, #2
```

Task 08- Step B

- Decode the following object code into assembly code:

```
00:      e3a00002

04:      eb000006

08:      e0811080

0c:      e2514005

10:      ca000000

14:      e0244004

18:      e3a00000

1c:      e3a07001

20:      ef000000

24:      e2000003

28:      e1a0f00e
```

```
1110 00 1 1101 0 0000 0000 0000 00000010
 AL     I  MOV S  Rn   Rd  #rot   #2      MOV r0, #2
1110 1011 0000 0000 0000 0000 0000 0110
```

Task 08- Step B

- Decode the following object code into assembly code:

```
00:     e3a00002
04:     eb000006
08:     e0811080
0c:     e2514005
10:     ca000000
14:     e0244004
18:     e3a00000
1c:     e3a07001
20:     ef000000
24:     e2000003
28:     e1a0f00e
```

```
1110 00 1 1101 0 0000 0000 0000 00000010
 AL     I  MOV S  Rn   Rd  #rot   #2      MOV r0, #2
1110 101 1 0000 0000 0000 0000 0000 0110
 AL    B L   offset
```

Task 08- Step B

- Decode the following object code into assembly code:

```
00:      e3a00002
04:      eb000006
08:      e0811080
0c:      e2514005
10:      ca000000
14:      e0244004
18:      e3a00000
1c:      e3a07001
20:      ef000000
24:      e2000003
28:      e1a0f00e
```

```
1110 00 1 1101 0 0000 0000 0000 00000010
 AL     I  MOV S  Rn   Rd  #rot   #2      MOV r0, #2
1110 101 1 0000 0000 0000 0000 0000 0110
 AL    B L jump addr = pc + offset*4 = 0x0c + 0x18
```

Task 08- Step B

- Decode the following object code into assembly code:

```
00:      e3a00002

04:      eb000006

08:      e0811080

0c:      e2514005

10:      ca000000

14:      e0244004

18:      e3a00000

1c:      e3a07001

20:      ef000000

24:      e2000003

28:      e1a0f00e
```

```
1110 00 1 1101 0 0000 0000 0000 00000010
 AL     I  MOV S   Rn  Rd  #rot   #2      MOV r0, #2
1110 101 1 0000 0000 0000 0000 0000 0110
 AL    B L jump addr = 0x24
```

Task 08- Step B

- Decode the following object code into assembly code:

```
00:      e3a00002

04:      eb000006

08:      e0811080

0c:      e2514005

10:      ca000000

14:      e0244004

18:      e3a00000

1c:      e3a07001

20:      ef000000

24:      e2000003

28:      e1a0f00e
```

```
1110 00 1 1101 0 0000 0000 0000 00000010
 AL     I  MOV S  Rn   Rd   #rot   #2      MOV r0, #2
1110 101 1 0000 0000 0000 0000 0000 0110
 AL    B L jump addr = 0x24                BL 0x24
```

Task 08- Step B

- Decode the following object code into assembly code:

```
00: e3a00002

04: eb000006

08: e0811080

0c: e2514005

10: ca000000

14: e0244004

18: e3a00000

1c: e3a07001

20: ef000000

24: e2000003

28: e1a0f00e
```

```
1110 00 1 1101 0 0000 0000 0000 00000010
 AL     I  MOV S  Rn    Rd  #rot   #2      MOV r0, #2
1110 101 1 0000 0000 0000 0000 0000 0110
 AL    B L jump addr = 0x24               BL 0x24
1110 0000 1000 0001 0001 0000 1000 0000
```

Task 08- Step B

- Decode the following object code into assembly code:

```
00: e3a00002

04: eb000006

08: e0811080

0c: e2514005

10: ca000000

14: e0244004

18: e3a00000

1c: e3a07001

20: ef000000

24: e2000003

28: e1a0f00e
```

```
1110 00 1 1101 0 0000 0000 0000 00000010
 AL     I  MOV S  Rn   Rd  #rot   #2      MOV r0, #2
1110 101 1 0000 0000 0000 0000 0000 0110
 AL    B L jump addr = 0x24               BL 0x24
1110 00 0 0100 0 0001 0001 00001 00 0 0000
 AL     I
```

Task 08- Step B

- Decode the following object code into assembly code:

```
00: e3a00002

04: eb000006

08: e0811080

0c: e2514005

10: ca000000

14: e0244004

18: e3a00000

1c: e3a07001

20: ef000000

24: e2000003

28: e1a0f00e
```

```
1110 00 1 1101 0 0000 0000 0000 00000010
 AL     I  MOV S  Rn   Rd  #rot   #2      MOV r0, #2
1110 101 1 0000 0000 0000 0000 0000 0110
 AL    B L jump addr = 0x24               BL 0x24
1110 00 0 0100 0 0001 0001 00001 00 0 0000
 AL     I  ADD S  Rn   Rd #shift LSL  Rm
```

Task 08- Step B

- Decode the following object code into assembly code:

```
00: e3a00002

04: eb000006

08: e0811080

0c: e2514005

10: ca000000

14: e0244004

18: e3a00000

1c: e3a07001

20: ef000000

24: e2000003

28: e1a0f00e
```

```
1110 00 1 1101 0 0000 0000 0000 00000010
 AL     I  MOV S  Rn   Rd  #rot   #2      MOV r0, #2
1110 101 1 0000 0000 0000 0000 0000 0110
 AL    B L jump addr = 0x24               BL 0x24
1110 00 0 0100 0 0001 0001 00001 00 0 0000
 AL     I  ADD S  Rn   Rd #shift LSL  Rm  ADD r1,r1,r0 #1




              .. Continue ....
```

Task 08- Step A

- Decode the following object code into assembly code:
  - Use Symbol Table

```
00: e28f401c

04: e5940000

08: e59f4018

0c: e0800004

10: e59f4014

14: e5843000

18: e3a07001

1c: e3a00000

20: ef000000

24: 000004d2
28: 12345678
2c: 00000000
```

```
1110 0010 1000 1111 0100 0000 0001 1100
```

Symbol Table

a	0x24
x	0x2c

Task 08- Step A

- Decode the following object code into assembly code:
  - Use Symbol Table

```
00: e28f401c

04: e5940000

08: e59f4018

0c: e0800004

10: e59f4014

14: e5843000

18: e3a07001

1c: e3a00000

20: ef000000

24: 000004d2
28: 12345678
2c: 00000000
```

```
1110 00 1 0100 0 1111 0100 0000 0001 1100
 AL     I  ADD S  Rn   Rd        #0x1c
```

Symbol Table

a	0x24
x	0x2c

Task 08- Step A

- Decode the following object code into assembly code:
  - Use Symbol Table

```
00: e28f401c

04: e5940000

08: e59f4018

0c: e0800004

10: e59f4014

14: e5843000

18: e3a07001

1c: e3a00000

20: ef000000

24: 000004d2
28: 12345678
2c: 00000000
```

```
1110 00 1 0100 0 1111 0100 0000 0001 1100
 AL     I  ADD S  Rn   Rd        #0x1c    ADD r4,pc,#0x1c
```

Symbol Table

a	0x24
x	0x2c

Task 08- Step A

- Decode the following object code into assembly code:
  - Use Symbol Table

```
00: e28f401c

04: e5940000

08: e59f4018

0c: e0800004

10: e59f4014

14: e5843000

18: e3a07001

1c: e3a00000

20: ef000000

24: 000004d2
28: 12345678
2c: 00000000
```

```
1110 00 1 0100 0 1111 0100 0000 0001 1100
 AL     I  ADD S  Rn   Rd        #0x1c    ADD r4,pc,#0x1c
                            addr = pc + 0x1c = 8 + 0x1c = 0x24
```

Symbol Table

a	0x24
x	0x2c

Task 08- Step A

- Decode the following object code into assembly code:
  - Use Symbol Table

```
00: e28f401c

04: e5940000

08: e59f4018

0c: e0800004

10: e59f4014

14: e5843000

18: e3a07001

1c: e3a00000

20: ef000000

24: 000004d2
28: 12345678
2c: 00000000
```

```
1110 00 1 0100 0 1111 0100 0000 0001 1100
 AL     I  ADD S  Rn   Rd        #0x1c    ADD r4,pc,#0x1c
                            addr = pc + 0x1c = 8 + 0x1c = 0x24
                                 = a (Symbol Table)
```

Symbol Table

a	0x24
x	0x2c

Task 08- Step A

- Decode the following object code into assembly code:
  - Use Symbol Table

```
00: e28f401c

04: e5940000

08: e59f4018

0c: e0800004

10: e59f4014

14: e5843000

18: e3a07001

1c: e3a00000

20: ef000000

24: 000004d2
28: 12345678
2c: 00000000
```

```
1110 00 1 0100 0 1111 0100 0000 0001 1100
 AL     I  ADD S  Rn   Rd        #0x1c    ADD r4,pc,#0x1c
                            addr = pc + 0x1c = 8 + 0x1c = 0x24
                                 = a (Symbol Table)
                                       → ADR r4, a
```

Symbol Table

a	0x24
x	0x2c

Task 08- Step A,C

- Decode the following object code into assembly code:
  - Use Symbol Table

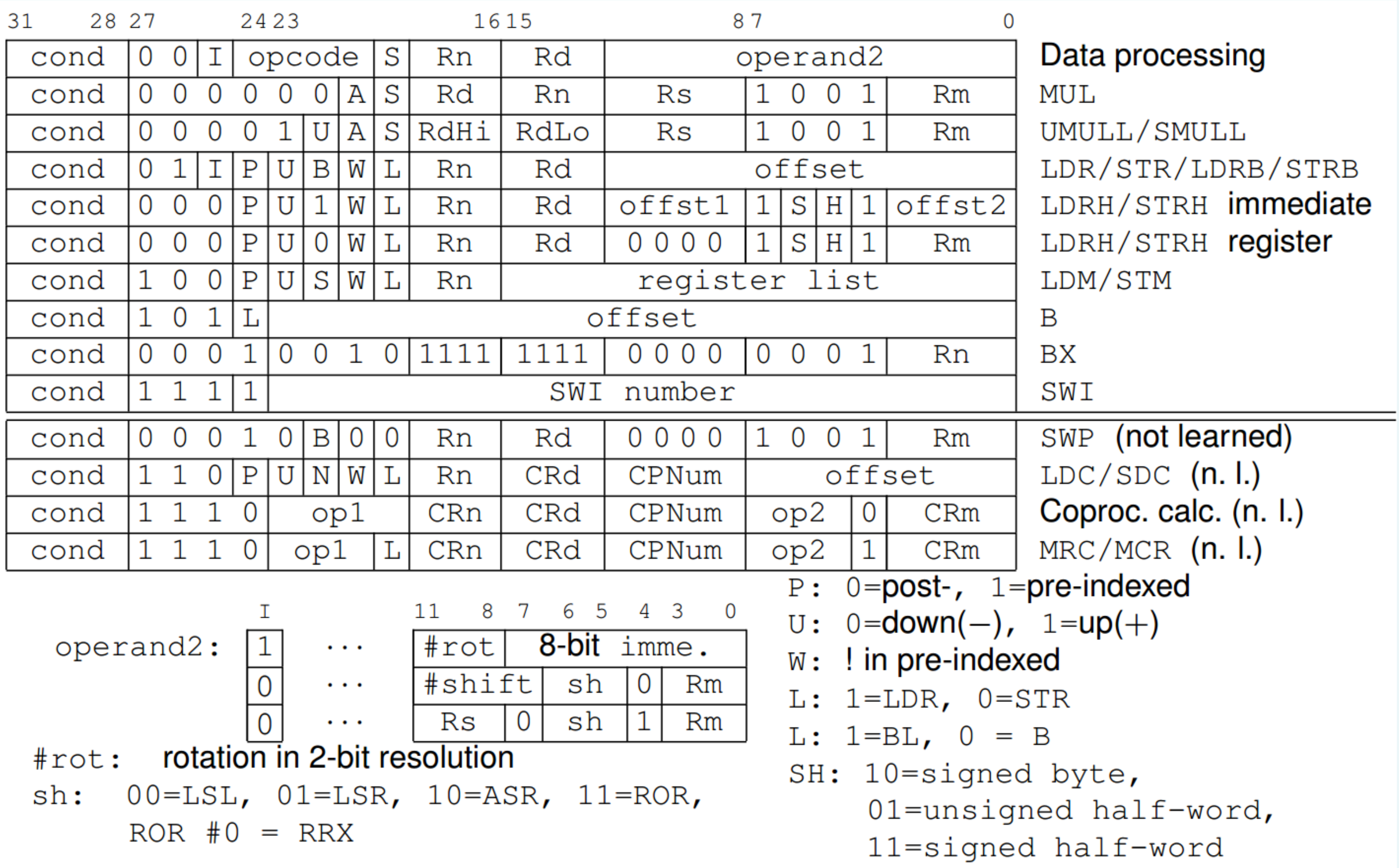
```
00:  e28f401c

04:  e5940000

08:  e59f4018

0c:  e0800004

10:  e59f4014

14:  e5843000

18:  e3a07001

1c:  e3a00000

20:  ef000000

24:  000004d2
28:  12345678
2c:  00000000
```

```
1110 00 1 0100 0 1111 0100 0000 0001 1100
 AL     I  ADD S  Rn   Rd        #0x1c    ADR r4, a





              ... Continue ...
```

Symbol Table

a	0x24
x	0x2c

ARM Instruction Set Format (I)



31 28	27			24	23				16	15	8	7				0	
cond	0	0	I	opcode				S	Rn	Rd	operand2						Data processing
cond	0	0	0	0	0	0	A	S	Rd	Rn	Rs	1	0	0	1	Rm	MUL
cond	0	0	0	0	1	U	A	S	RdHi	RdLo	Rs	1	0	0	1	Rm	UMULL/SMULL
cond	0	1	I	P	U	B	W	L	Rn	Rd	offset						LDR/STR/LDRB/STRB
cond	0	0	0	P	U	1	W	L	Rn	Rd	offst1	1	S	H	1	offst2	LDRH/STRH immediate
cond	0	0	0	P	U	0	W	L	Rn	Rd	0 0 0 0	1	S	H	1	Rm	LDRH/STRH register
cond	1	0	0	P	U	S	W	L	Rn	register list							LDM/STM
cond	1	0	1	L	offset												B
cond	0	0	0	1	0	0	1	0	1111	1111	0 0 0 0	0	0	0	1	Rn	BX
cond	1	1	1	1	SWI number												SWI
cond	0	0	0	1	0	B	0	0	Rn	Rd	0 0 0 0	1	0	0	1	Rm	SWP (not learned)
cond	1	1	0	P	U	N	W	L	Rn	CRd	CPNum	offset					LDC/SDC (n. l.)
cond	1	1	1	0	op1				CRn	CRd	CPNum	op2			0	CRm	Coproc. calc. (n. l.)
cond	1	1	1	0	op1			L	CRn	CRd	CPNum	op2			1	CRm	MRC/MCR (n. l.)

operand2:

I		11 8	7	6 5	4	3 0
1	···	#rot	8-bit imme.			
0	···	#shift		sh	0	Rm
0	···	Rs	0	sh	1	Rm

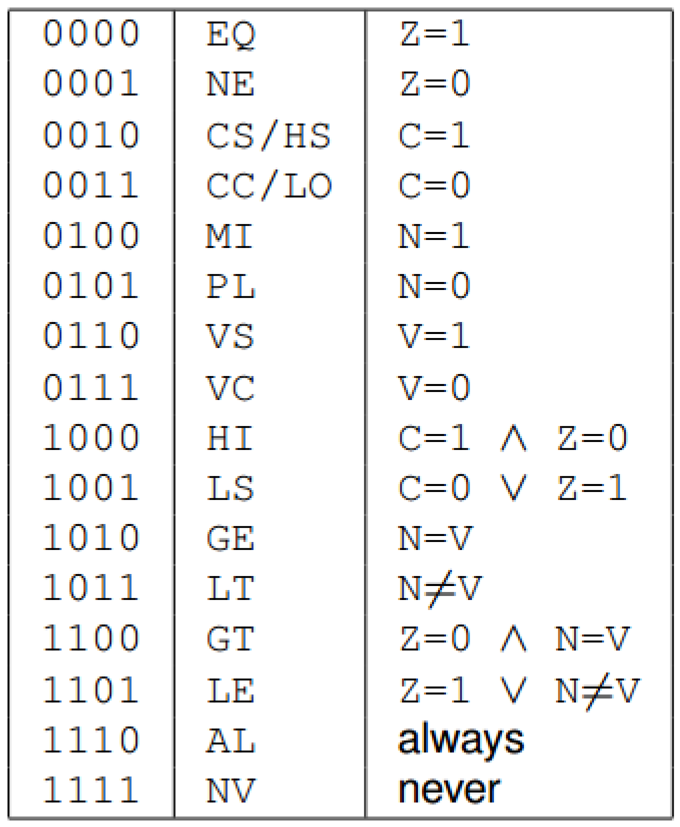
#rot: rotation in 2-bit resolution

sh: 00=LSL, 01=LSR, 10=ASR, 11=ROR,
ROR #0 = RRX

P: 0=post-, 1=pre-indexed
U: 0=down(−), 1=up(+)
W: ! in pre-indexed
L: 1=LDR, 0=STR
L: 1=BL, 0 = B
SH: 10=signed byte,
01=unsigned half-word,
11=signed half-word

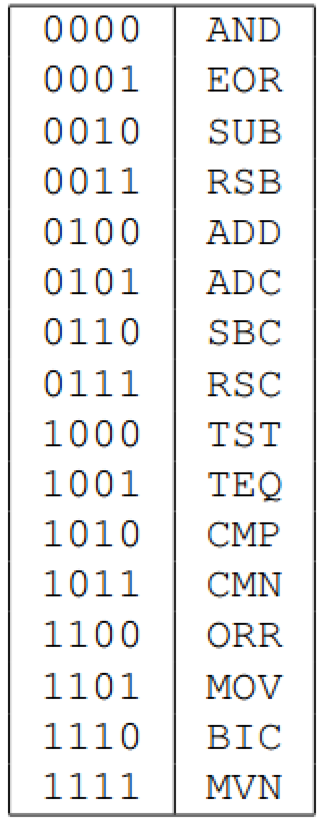
ARM Instruction Set Format (II)

- cond



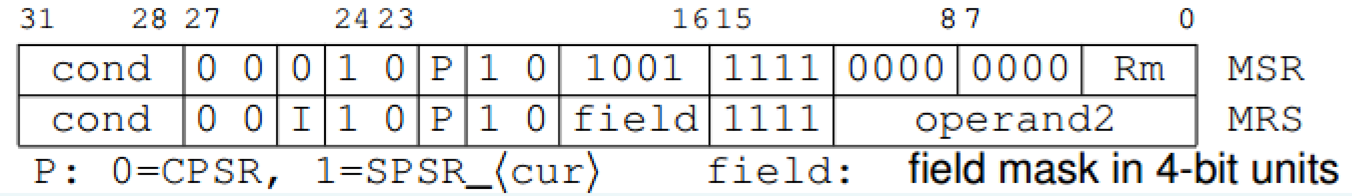
0000	EQ	Z=1
0001	NE	Z=0
0010	CS/HS	C=1
0011	CC/LO	C=0
0100	MI	N=1
0101	PL	N=0
0110	VS	V=1
0111	VC	V=0
1000	HI	C=1 ∧ Z=0
1001	LS	C=0 ∨ Z=1
1010	GE	N=V
1011	LT	N≠V
1100	GT	Z=0 ∧ N=V
1101	LE	Z=1 ∨ N≠V
1110	AL	always
1111	NV	never

- opcode



0000	AND
0001	EOR
0010	SUB
0011	RSB
0100	ADD
0101	ADC
0110	SBC
0111	RSC
1000	TST
1001	TEQ
1010	CMP
1011	CMN
1100	ORR
1101	MOV
1110	BIC
1111	MVN

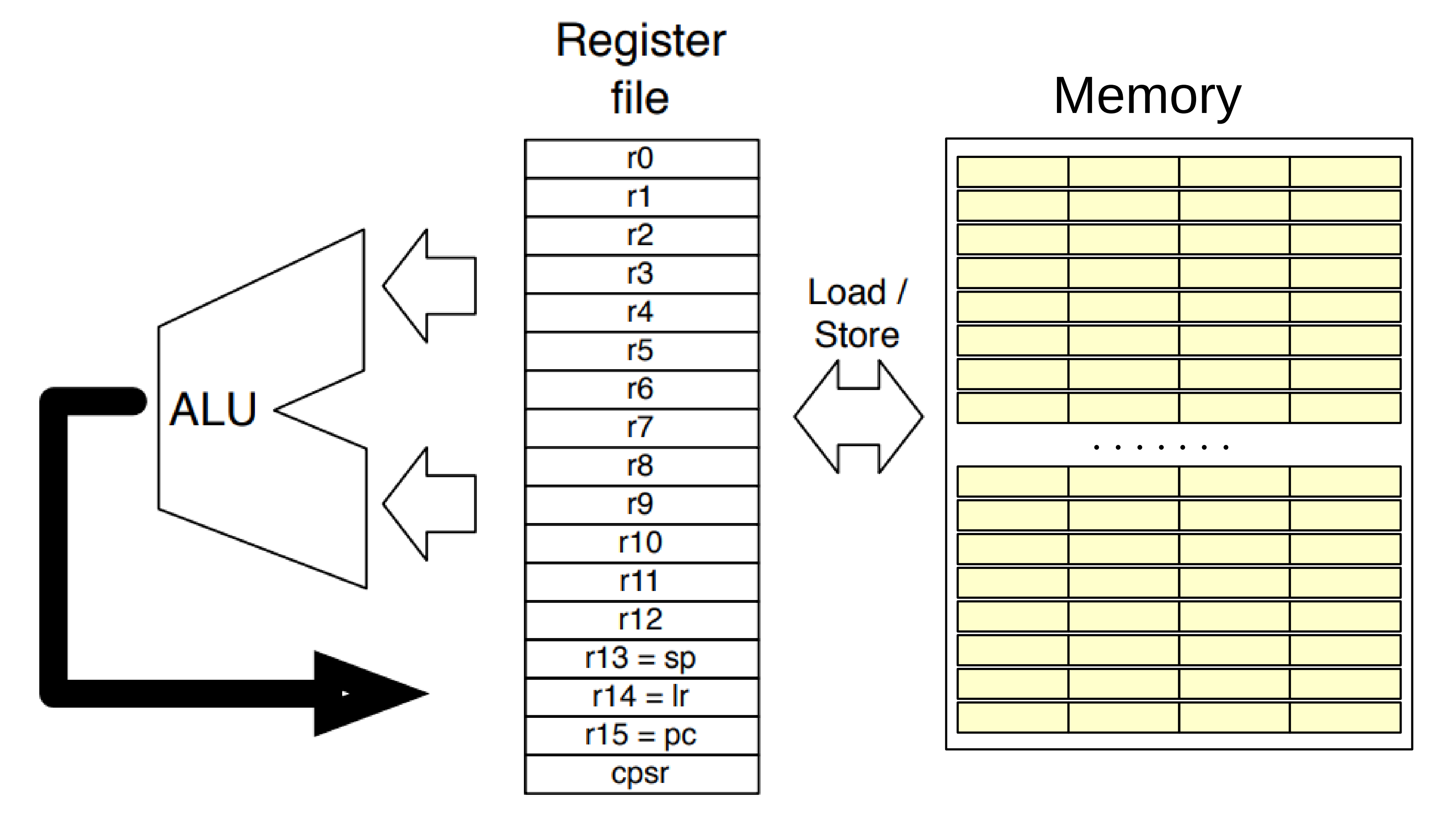
- MSR / MRS : special case of data processing



31 28	27		24	23			16	15		8 7	0	
cond	0 0	0	1 0	P	1 0	1001	1111	0000	0000	Rm		MSR
cond	0 0	I	1 0	P	1 0	field	1111	operand2				MRS

P: 0=CPSR, 1=SPSR_⟨cur⟩ field: field mask in 4-bit units

Simplified View of ARM Computer in User-Mode


Register file
Memory
r0
r1
r2
r3
r4
r5
r6
r7
r8
r9
r10
r11
r12
r13 = sp
r14 = lr
r15 = pc
cpsr
ALU
Load / Store
. . . .