ARM Assembly

Young W. Lim

June 8, 2016

ARM Assembly Categories

- Arithmetic
- Data Transfer
- Logical
- Conditional Branch
- Unconditional Branch

ARM Assembly Category Instructions

- Arithmetic :
  add, subtract
- Data Transfer :
  load/store (halfword / halfword / Byte) (signed),
  swap, mov
- Logical :
  and, or, not,
  logical shift (right/left)
- Conditional Branch :
  compare, branch on conditions
- Unconditional Branch :
  brach, branch and link

Arithmetic Instructions

```
add r1, r2, r3      ; r1 = r2 + r3
sub r1, r2, r3      ; r1 = r2 - r3
```

Data Transfer Instructions

```
ldr   r1, [r2, #20]     ; r1 = M[r2+20]
ldrh  r1, [r2, #20]     ; r1 = M[r2+20], halfword
ldrhs r1, [r2, #20]     ; r1 = M[r2+20], halfword, signed
ldrb  r1, [r2, #20]     ; r1 = M[r2+20]
ldrbs r1, [r2, #20]     ; r1 = r2 + r3
str   r1, [r2, #20]     ; M[r2+20] = r1
strh  r1, [r2, #20]     ; M[r2+20] = r1, halfword

swap  r1, [r2, #20]     ; r1 <-> M[r2+20]
mov   r1, r2            ; r1 = r2
```

Logical Instructions

```
and   r1, r2, r3     ; r1 = r2 & r3
orr   r1, r2, r3     ; r1 = r2 | r3

mvn   r1, r2         ; r1 = ~r2

lsl   r1, r2, #10    ; r1 = r2 << 10
lsr   r1, r2, #10    ; r1 = r2 >> 10
```

Conditional Branch Instructions

```
cmp   r1, r2          ; flag set by r1-r2
beq   25              ; goto PC+8+(25*4) if Z set

eq, ne, lt, le, gt, ge, lo, ls, hi, hs, vs, vc, mi, pl
```

Unconditional Branch Instructions

```
b       25              ; goto PC+8+(25*4)
bl      25              ; r14=PC+4, goto PC+8+(25*4)
```

Summary 1

```
ADC     ADD with Carry (Rd = Rn + Op2 + Carry)
ADD     ADD (Rd = Rn + Op2)
AND     AND (Rd = Rn & Op2)
B       Branch (R15 = address)
BIC     BIt Clear (Rd = Rn & ~Op2)
BL      Branch and Link (R14 = R15, R15 = address)
BX      Branch and Exchange (R15 = Rn, T bit  = Rn[0])
CDP     Coprocessor Data Processing
CMN     Compare Negative (CPSR flags = Rn + Op2)
CMP     Compare (CPSR flags = Rn - Op2)
EOR     Exclusive OR (Rd = Rn ^ Op2)
LDC     Load Coprocessor from memory
LDM     Load Multiple Registers (Pop the stack)
LDR     Load Register from memory (Rd = address)
```

Summary 2

```
MCR     Move CPU reg to Coproc reg (cRn = Rn {<op>cRm}
MLA     Multiply Accumulate (Rd = (Rm * Rs) + Rn)MOV     Move reg
MRC     Move from Coproc reg to CPU reg (Rn = cRn {<op>cRm})
MRS     Move PSR status/flags to register (Rn = PSR)
MSR     Move register to PSR status/flags (PSR = Rm)
MUL     Multiply (Rd = Rm * Rs)
MVN     Move Negative Register (Rd = 0xFFFFFFFF ^ Op2)
ORR     OR (Rd = rn | Op2)
RSB     Reverse Subtract (Rd = Op2 - Rn)
RSC     Reverse Subtract with Carry (Rd = Op2 - Rn -1 +Carry)
SBC     Subtract with Carry (Rd = Rn - Op2 -1 +Carry)
```

Summary 3

```
STC     Store Coproc reg to memory (address = cRn)
STM     Store Multiple (Push the stack)
STR     Store register to memory (<address> = Rd)
SUB     Subtract (Rd = Rn - Op2)
SWI     Software Interrupt (OS system call)
SWP     Swap register with memory (Rd <-> [Rn])
TEQ     Test bitwise equality (CPSR flags = Rn ^ Op2)
TST Test bits (CPSR flags = Rn & Op2)
```

Addressing Modes

- Immediate/Register/Sacled Register Offset
- Scaled Register Offset
- Immediate Pre/Post-Indexed
- Register Pre/Post-Indexed
- Scaled Register Pre-Indexed
- Immediate
- Register
- Scaled Register
- PC-relative Addressing

Cond Field

- EQ (EQaul)
- NE (Not Equal)
- HS (unsigned Higher or Same)
- LO (unsigend LOwer)
- MI (Minus, <0)
- PL (PLus, >0)
- VS (oVerflow Set, overflow)
- VC (oVerflow Clear, no overflow)
- HI (unsigned HIgher)
- LS (unsinged Lower or Same)
- GE (signed Greater than or Equal)
- LT (signed Less Than)
- GT (signed Greater Than)
- LE (signed Less than or Equal)
- AL (ALways)
- NV (reserved)

Across Procedure Calls

- Preserved
  - Variable registers : r4-r11
  - Stack pointer register : sp
  - Link register : lr
  - Stack above the stack pointer
- Not preserved
  - Argument registers : r0-r3
  - Intra-procedure-call scratch register : r12
  - Stack below the stack pointer

ABI Register Conventions

- a1-a2 : 0-1, Argument / return result / scratch register, changing
- a3-a4 : 2-3, Argument / scratch register, changing
- v1-v8 : 4-11, Variables for local routine, preserved
- ip: 12, Intra-procedure call scratch register, changing
- sp, 13, Stack pointer, preserved
- lr, 14, Link register (return address), preserved
- pc, 15, Program counter, N.A

Reference

[1] D. Harris, Digital Design and Computer Architecture", 2nd ed.
[2] D.A. Patterson & J.H. Hennessy, Computer Organization and Design (ARM ed)