ARM programming

Tips

A general source line is:

`[label] [operation] [;comment]`

- Labels must start at the beggining of lines
- Operations must be preceeded by white space(s)

Architecture

Registers

- R0-R12 general
- R13 sp
- R14 lr
- R15 pc

APSR

- N: negative result
- Z: zero result
- C: carry out (over 32 bits)
- V: overflow (over 31 bits)

Conditional suffixes (see others in slides)

- EQ (if Z set)
- NE (if Z clear)
- GE (>= Ns and Vs or Nc and Vc)
- LE (<= Ns and Vc or Nc and Vs)
- GT (>)
- LT (<)
- LS (C clear or Z, uns low or same)
- MI (N set, negative)
- VS (V set, overflow)
- VC (V clear, no overflow)

ALU instructions

Arithmetic instructions

- CMP <Rd> <op> (sub and update flags)
- CMN <Rd> <op> (add and update flags)
- TST <Rd> <op> (logical AND, update)
- TEQ <Rd> <op> (equivalence, update)
- MSR <Sr> <R> (move special to reg)
- MRS <R> <Sr> (move reg to special)
- ADD <Rd>, <Rs>, <Rs>
- SUB <Rd>, <Rs>, <Rs>
- RSB <Rd>, <Rn>, <op> (can shift <op>)

  example: RSB r0, r1, r2, LSL #2
- MUL <Rd>, <Rn>, <Rm>
- UMULL <Rd1>, <Rd2>, <Rn>, <Rm>

  SMULL <Rd1>, <Rd2>, <Rn>, <Rm>

  (64 bit MUL)
- MLA <Rd>, <Rn>, <Rm>, <Ra>

  (Rd = Rn * Rm + Ra)
- MLS <Rd>, <Rn>, <Rm>, <Ra>

  (Rd = Rn * Rm - Ra)

- UMLAL/SMLAL <Rd1>, <Rd2>, <Rm>, <Rn>

  (Rd1, Rd2 = Rn * Rm + Rd1, Rd2)

Logic instructions

- AND <Rd>, <Rd>, op
- BIC <Rd>, <Rn>, op (AND NOT)
- ORR <Rd>, <Rn>, op
- EOR <Rd>, <Rn>, op (XOR)
- ORN <Rd>, <Rn>, op (OR NOT)
- MVN <Rd>, <Rn> (NOT)

Shift instructions

- LSL <Rd>, <Rn>, op
- LSR <Rd>, <Rn>, op
- A <Rd>, <Rn>, op (reinsert MSB)
- ROR <Rd>, <Rn>, op (rotate)
- RRX <Rd>, <Rn> (rotate)

Directives

- AREA sectionName [,attr(s)]

(Define CODE or DATA block; if sectionName starts with a number must be enclosed in vert bars. Attributes may be CODE, READONLY, DATA, READWRITE, ALIGN=EXPR).

- name RN regIndex (rename register)
- name EQU expr (create numeric constant)
- ENTRY (start program)
- label DCB/DCW/DCD data (allocate memory, optional U=unaligned)
- ALIGN num (align code/data to boundary)
- label SPACE bytes (zeroed block of mem)
- LTORG (start literal pool)
- END (end of source file)

Literal pool and constants

- MOV <Rd>, <src> (src may be a register or a constant value) if <src> is a register can add an optional [shift] (e.g. LSL #n) MOVW is the same but with 16-bit values.
- MOVT <Rd>, #<constant> (move i n the high halfword)
- LDR <Rd>, =<constant> (may create literal pool)

Accessing memory

Load

- LDR/LDRB/LDRH/LDRSB/LDRSH/LDRD/LDM

Store

- STR/STRB/STRH/STRD/STM

D = double, M = multiple

Addressing modes

- l/s <Rd>, [<Rn>,<offset>]{!}

(pre-indexed: offset may be 12-bit constant or 3-shifted register; ! indicates update at the end of instruction)

- l/s <Rd>, [<Rn>], <offset>

(post-indexed: <Rn> is updated adding offset; Rn is always updated)

Branches

- B/BX <label>/<Rn>
- BL/BLX <label>/<Rn> (link in R14=LR)

stop B stop (infinite loop, idle)

(immediate value is 24-bit, with BX can jump up to 4GB)

- CBZ <Rn>, <label> (jump if Rn=0)
- CBNZ <Rn>, <label> (jump if Rn!=0)

(Rn must be among R0-R7; CBZ doesn’t set flags, jumps only forward, shorter range)