ARM assembly language reference card

MOVdS reg, arg copy argument (S = set flags) \(B_{cd}\) \(imm_{12}\) branch to \(imm_{12}\) words away
MVNdS reg, arg copy bitwise NOT of argument \(B_{ld}\) \(imm_{12}\) copy PC to LR, then branch
ANDdS reg, reg, arg bitwise AND \(BX_{cd}\) reg copy reg to PC
ORRedS reg, reg, arg bitwise OR \(SW_{ld}\) \(imm_{24}\) software interrupt
EOEdS reg, reg, arg bitwise exclusive-OR \(LDR_{dB}\) reg, mem loads word/byte from memory
BICdS reg, reg, arg, \(arg_{b}\) bitwise exclusive-OR AND \(( NOT \ arg_{b})\) \(STR_{dB}\) reg, mem stores word/byte to memory
ADdS reg, reg, arg add \(LDM_{dum}\ reg\), \(mreg\) loads into multiple registers
SUBdS reg, reg, arg subtract \(STM_{dum}\ reg\), \(mreg\) stores multiple registers
RSBdS reg, reg, arg subtract reversed arguments \(SWP_{dB}\ reg_{d}, reg_{m}, [reg_{n}]\) copies \(reg_{m}\) to memory at \(reg_{n}\), old value at address \(reg_{n}\) to \(reg_{d}\)
ADCdS reg, reg, arg add with carry flag
SBCdS reg, reg, arg subtract with carry flag
RSCdS reg, reg, arg reverse subtract with carry flag
CMPd reg, arg update flags based on subtraction
CMNd reg, arg update flags based on addition
TSTd reg, arg update flags based on bitwise AND
TEQd reg, arg update flags based on bitwise exclusive-OR

\underline{reg}: register

- \(R_0\) to \(R_{15}\) register according to number
- \(SP\) register 13
- \(LR\) register 14
- \(PC\) register 15

\underline{um}: update mode

- \(IA\) increment, starting from \(reg\)
- \(IB\) increment, starting from \(reg + 4\)
- \(DA\) decrement, starting from \(reg\)
- \(DB\) decrement, starting from \(reg - 4\)

\underline{cd}: condition code

- \(AL\) or omitted always
- \(EQ\) equal (zero)
- \(NE\) nonequal (nonzero)
- \(CS\) carry set (same as HS)
- \(CC\) carry clear (same as LO)
- \(MI\) minus
- \(PL\) positive or zero
- \(VS\) overflow set
- \(VC\) overflow clear
- \(HS\) unsigned higher or same
- \(LO\) unsigned lower
- \(HI\) unsigned higher
- \(LS\) unsigned lower or same
- \(GE\) signed greater than or equal
- \(LT\) signed less than
- \(GT\) signed greater than
- \(LE\) signed less than or equal

\underline{arg}: right-hand argument

- \(#imm_{8},\) immediate (rotated into 8 bits)
- \(reg\) register
- \(reg, shift\) register shifted by distance

\underline{mem}: memory address

- \([reg, \# imm_{12}]\) reg offset by constant
- \([reg, \pm reg]\) reg offset by variable bytes
- \([reg_{a}, \pm reg, shift]\) \(reg_{a}\) offset by shifted variable \(reg_{b}^{†}\)
- \([reg, \# imm_{12}]\) \(reg\) offset by constant, then access memory
- \([reg, \pm reg]\) \(reg\) offset by variable bytes, access memory
- \([reg, \pm reg, shift]\) \(reg\) offset by shifted variable\(^{†}\), access memory
- \([reg], \# imm_{12}\) access address \(reg\), then update \(reg\) by offset
- \([reg], \pm reg\) access address \(reg\), then update \(reg\) by variable
- \([reg], \pm reg, shift\) access address \(reg\), update \(reg\) by shifted variable\(^{†}\)

\(†\) shift distance must be by constant

\underline{shift}: shift register value

- \(LSL \# imm_{5}\) shift left 0 to 31
- \(LSR \# imm_{5}\) logical shift right 1 to 32
- \(ASR \# imm_{5}\) arithmetic shift right 1 to 32
- \(ROR \# imm_{5}\) rotate right 1 to 31
- \(ROR\) rotate carry bit into top bit
- \(LSL reg\) shift left by register
- \(LSR reg\) logical shift right by register
- \(ASR reg\) arithmetic shift right by register
- \(ROR reg\) rotate right by register