ARM Assembly

Memory access instructions

```
LDR     Rd, [Rn]          ; load 32-bit number at [Rn] to Rd
LDR     Rd, [Rn,#off]     ; load 32-bit number at [Rn+off] to Rd
LDR     Rd, =value        ; set Rd equal to any 32-bit value (PC rel)
LDRH    Rd, [Rn]          ; load unsigned 16-bit at [Rn] to Rd
LDRH    Rd, [Rn,#off]     ; load unsigned 16-bit at [Rn+off] to Rd
LDRSH   Rd, [Rn]          ; load signed 16-bit at [Rn] to Rd
LDRSH   Rd, [Rn,#off]     ; load signed 16-bit at [Rn+off] to Rd
LDRB    Rd, [Rn]          ; load unsigned 8-bit at [Rn] to Rd
LDRB    Rd, [Rn,#off]     ; load unsigned 8-bit at [Rn+off] to Rd
LDRSB   Rd, [Rn]          ; load signed 8-bit at [Rn] to Rd
LDRSB   Rd, [Rn,#off]     ; load signed 8-bit at [Rn+off] to Rd
STR     Rt, [Rn]          ; store 32-bit Rt to [Rn]
STR     Rt, [Rn,#off]     ; store 32-bit Rt to [Rn+off]
STRH    Rt, [Rn]          ; store least sig. 16-bit Rt to [Rn]
STRH    Rt, [Rn,#off]     ; store least sig. 16-bit Rt to [Rn+off]
STRB    Rt, [Rn]          ; store least sig. 8-bit Rt to [Rn]
STRB    Rt, [Rn,#off]     ; store least sig. 8-bit Rt to [Rn+off]
PUSH    {Rt}              ; push 32-bit Rt onto stack
POP     {Rd}              ; pop 32-bit number from stack into Rd
ADR     Rd, label         ; set Rd equal to the address at label
MOV     {S} Rd, <op2>     ; set Rd equal to op2
MOV     Rd, #im16         ; set Rd equal to im16, im16 is 0 to 65535
MVN     {S} Rd, <op2>     ; set Rd equal to -op2
```

Interrupt instructions

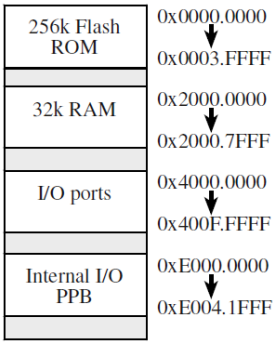
```
CPSIE   I                 ; enable interrupts (I=0)
CPSID   I                 ; disable interrupts (I=1)
```

Arithmetic instructions

```
ADD     {S} {Rd,} Rn, <op2>   ; Rd = Rn + op2
ADD     {S} {Rd,} Rn, #im12   ; Rd = Rn + im12, im12 is 0 to 4095
SUB     {S} {Rd,} Rn, <op2>   ; Rd = Rn - op2
SUB     {S} {Rd,} Rn, #im12   ; Rd = Rn - im12, im12 is 0 to 4095
RSB     {S} {Rd,} Rn, <op2>   ; Rd = op2 - Rn
RSB     {S} {Rd,} Rn, #im12   ; Rd = im12 – Rn
CMP     Rn, <op2>             ; Rn – op2 sets the NZVC bits
CMN     Rn, <op2>             ; Rn - (-op2) sets the NZVC bits
MUL     {S} {Rd,} Rn, Rm      ; Rd = Rn * Rm signed or unsigned
MLA     Rd, Rn, Rm, Ra        ; Rd = Ra + Rn*Rm signed or unsigned
MLS     Rd, Rn, Rm, Ra        ; Rd = Ra - Rn*Rm signed or unsigned
UDIV    {Rd,} Rn, Rm          ; Rd = Rn/Rm unsigned
SDIV    {Rd,} Rn, Rm          ; Rd = Rn/Rm signed
```

Examples of flexible operand <op2> creating the 32-bit number.
E.g., Rd = Rn+op2

```
ADD Rd, Rn, Rm            ; op2 = Rm
ADD Rd, Rn, Rm, LSL #n    ; op2 = Rm<<n Rm is signed, unsigned
ADD Rd, Rn, Rm, LSR #n    ; op2 = Rm>>n Rm is unsigned
ADD Rd, Rn, Rm, ASR #n    ; op2 = Rm>>n Rm is signed
ADD Rd, Rn, #constant     ; op2 = constant
```



Branch instructions

```
B       label     ; branch to            label Always
BEQ     label     ; branch if Z == 1 Equal
BNE     label     ; branch if Z == 0 Not equal
BCS     label     ; branch if C == 1 Higher or same, unsigned ≥
BHS     label     ; branch if C == 1 Higher or same, unsigned ≥
BCC     label     ; branch if C == 0 Lower, unsigned <
BLO     label     ; branch if C == 0 Lower, unsigned <
BMI     label     ; branch if N == 1 Negative
BPL     label     ; branch if N == 0 Positive or zero
BVS     label     ; branch if V == 1 Overflow
BVC     label     ; branch if V == 0 No overflow
BHI     label     ; branch if C==1 and Z==0 Higher, unsigned >
BLS     label     ; branch if C==0 or Z==1 Lower or same, unsigned ≤
BGE     label     ; branch if N == V Greater than or equal, signed ≥
BLT     label     ; branch if N != V Less than, signed <
BGT     label     ; branch if Z==0 and N==V Greater than, signed >
BLE     label     ; branch if Z==1 or N!=V Less than or equal, signed ≤
BX      Rm        ; branch indirect to location specified by Rm
BL      label     ; branch to subroutine at      label
BLX     Rm        ; branch to subroutine indirect specified by Rm
```

Logical instructions

```
AND     {S} {Rd,} Rn, <op2>       ; Rd=Rn&op2 (op2 is 32 bits)
ORR     {S} {Rd,} Rn, <op2>       ; Rd=Rn|op2 (op2 is 32 bits)
EOR     {S} {Rd,} Rn, <op2>       ; Rd=Rn^op2 (op2 is 32 bits)
BIC     {S} {Rd,} Rn, <op2>       ; Rd=Rn&(~op2) (op2 is 32 bits)
ORN     {S} {Rd,} Rn, <op2>       ; Rd=Rn|(~op2) (op2 is 32 bits)
LSR     {S} Rd, Rm, Rs    ; logical shift right Rd=Rm>>Rs (unsigned)
LSR     {S} Rd, Rm, #n    ; logical shift right Rd=Rm>>n (unsigned)
ASR     {S} Rd, Rm, Rs    ; arithmetic shift right Rd=Rm>>Rs (signed)
ASR     {S} Rd, Rm, #n    ; arithmetic shift right Rd=Rm>>n (signed)
LSL     {S} Rd, Rm, Rs    ; shift left Rd=Rm<<Rs (signed, unsigned)
LSL     {S} Rd, Rm, #n    ; shift left Rd=Rm<<n (signed, unsigned)
```

Notes Ra Rd Rm Rn Rt represent 32-bit registers

value	any 32-bit value: signed, unsigned, or address
{S}	if S is present, instruction will set condition codes
#im12	any value from 0 to 4095
#im16	any value from 0 to 65535
{Rd,}	if Rd is present Rd is destination, otherwise Rn
#n	any value from 0 to 31
#off	any value from -255 to 4095
label	any address within the ROM of the microcontroller
op2	the value generated by <op2>