Functions (8A)

Young Won Lim
7/21/20

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using LibreOffice.

Based on

ARM System-on-Chip Architecture, 2nd ed, Steve Furber

Introduction to ARM Cortex-M Microcontrollers
– Embedded Systems, Jonathan W. Valvano

Digital Design and Computer Architecture,
D. M. Harris and S. L. Harris

https://thinkingeek.com/arm-assembler-raspberry-pi/

Supporting Procedures

1. put parameters in a place where the procedure can access them
2. transfer control to the procedure
3. acquire the storage resources needed fr the procedure
4. perform the desired task
5. put the result value in a place where the calling program can access it
6. return control to the points of origin, since a procedure can be called from several points in a program

ARM System-on-Chip Architecture, 2nd ed, Steve Furber

Registers

R0, R1, R2, R3 : four argument registers to pass parameters

LR : one link register containing the return address register
to the point of origin

Introduction to ARM Cortex-M Microcontrollers – Embedded Systems, Jonathan W. Valvano

Instructions for procedures

```
BL   ProcedureAddress
```

jumps to an address and simultaneously saves
the address of the following instruction in register LR

```
MOV     PC, LR
```

Introduction to ARM Cortex-M Microcontrollers – Embedded Systems, Jonathan W. Valvano

Instructions for procedures

```
B{cond}     label   ; branch to label
BX{cond}    Rm      ; branch indirect to location specified by Rm
BL{cond}    label   ; branch to subroutine at label
BLX{cond}   Rm      ; branch to subroutine indirect specified by Rm
```

Introduction to ARM Cortex-M Microcontrollers – Embedded Systems, Jonathan W. Valvano

Instructions for procedures

```
uint32_t Num;

void Change(void) {
        Num = Num + 25;
}

void main(void) {
        Num = 0;
        while (1) {
                Change();
        }
}
```

Introduction to ARM Cortex-M Microcontrollers – Embedded Systems, Jonathan W. Valvano

Instructions for procedures

```
Change  LDR     R1, =Num        ; 5) R1 = &Num
        LDR     R0, [R1]        ; 6) R0 = Num
        ADD     R0, R0, #25     ; 7) R0 = Num + 25
        STR     R0, [R1]        ; 8) Num = Num + 25
        BX      LR              ; 9) return

Main    LDR     R1, =Num        ; 1) R1 = &Num
        MOV     R0, #0          ; 2) R0 = 0
        STR     R0, [R1]        ; 3) Num = 0
Loop    BL      Change          ; 4) call to Change
        B       Loop            ; 10) repeat
```

Introduction to ARM Cortex-M Microcontrollers – Embedded Systems, Jonathan W. Valvano

Instructions for procedures

```
uint32_t Num;

void Change(void) {
        if (Num <25600) {
                Num = Num + 25;
        }
}

void main(void) {
        Num = 0;
        while (1) {
                Change();
        }
}
```

Introduction to ARM Cortex-M Microcontrollers – Embedded Systems, Jonathan W. Valvano

Instructions for procedures

```
Change  LDR     R1, =Num        ; R1 = &Num
        LDR     R0, [R1]        ; R0 = Num
        CMP     R0, #25600      ;
        BHS     skip
        ADD     R0, R0, #25     ; R0 = Num + 25
        STR     R0, [R1]        ; Num = Num + 25
Skip    BX      LR              ; return

Main    LDR     R1, =Num        ; R1 = &Num
        MOV     R0, #0          ; R0 = 0
        STR     R0, [R1]        ; Num = 0
Loop    BL      Change          ; call to Change
        B       Loop            ; repeat
```

Introduction to ARM Cortex-M Microcontrollers – Embedded Systems, Jonathan W. Valvano

Instructions for procedures

```
uint32_t Num;

void Change(void) {
        if (Num <100) {
                Num = Num + 1;
        } else {
                Num = -100;
        }
}

void main(void) {
        Num = 0;
        while (1) {
                Change();
        }
}
```

Introduction to ARM Cortex-M Microcontrollers – Embedded Systems, Jonathan W. Valvano

Instructions for procedures

```
Change  LDR     R1, =Num            ; R1 = &Num
        LDR     R0, [R1]            ; R0 = Num
        CMP     R0, #100            ;
        BGE     else
        ADD     R0, R0, #1          ; R0 = Num + 1
        B       skip
Else    MOV     R0, #-100           ; R0 = -100
skip    STR     R0, [R1]            ; Num = Num + 1 or -100
        BX      LR                  ; return

Main    LDR     R1, =Num            ; R1 = &Num
        MOV     R0, #0              ; R0 = 0
        STR     R0, [R1]            ; Num = 0
Loop    BL      Change              ; call to Change
        B       Loop                ; repeat
```

Introduction to ARM Cortex-M Microcontrollers – Embedded Systems, Jonathan W. Valvano

Pointer access to an array

References

[1] ftp://ftp.geoinfo.tuwien.ac.at/navratil/HaskellTutorial.pdf

[2] https://www.umiacs.umd.edu/~hal/docs/daume02yaht.pdf