Exam 1

Date: Oct 3, 2019

UT EID: _____   Professor: Valvano

Printed Name: _____   _____
                          Last,   First

Your signature is your promise that you have not cheated and will not cheat on this exam, nor will you help others to cheat on this exam:

Signature: _____

Instructions:

- Closed book and closed notes. No books, no papers, no data sheets (other than the last two pages of this Exam)
- No devices other than pencil, pen, eraser (no calculators, no electronic devices), please turn cell phones off.
- Please be sure that your answers to all questions (and all supporting work that is required) are contained in the space (boxes) provided. *Anything outside the boxes/blanks will be ignored in grading.* You may use the back of the sheets for scratch work.
- You have 75 minutes, so allocate your time accordingly.
- For all questions, unless otherwise stated, find the most efficient (time, resources) solution.
- Unless otherwise stated, make all I/O accesses friendly and all subroutines AAPCS compliant.
- *Please read the entire exam before starting.*

(12) Question 1.
(2) Part a) Which equation describes the power dissipated in a resistor? .....
A) P = V/I   B) P = V²*R
C) P = I²*R   D) P = V/R²
E) Some of A – D   F) None of A – D

(2) Part b) If you add an *n*-bit signed number to an *m*-bit signed number, what is the maximum number of bits in the sum? Assume $n \geq m$. ....

(2) Part c) Assume you add two unsigned 32-bit numbers using the `ADDS R2,R1,R0` instruction. Which bit is set on overflow? ....

(2) Part d) Sketch an approximate plot of LED current as a function of LED voltage (show plot for both positive and negative voltages).
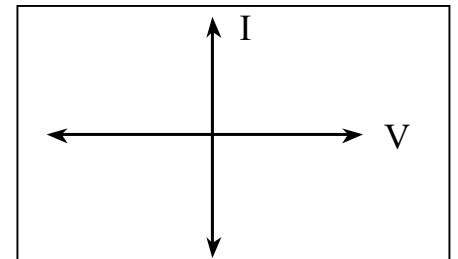
For Parts e and f) choose
      A) Restricted to within the function
      B) Stored permanently in ROM
      C) Stored permanently in RAM
      D) Nonvolatile
      E) Stored in R14
      F) Stored temporarily in a register or on the stack
      G) Can be accessed from anywhere in the system

(2) Part e) What is the scope of a local variable in C? ....

(2) Part f) Where are global variables allocated in C? ....

(6) Question 2. Consider a ULN2003B driver chip, where the input **In** is connected to B, the emitter is grounded, and the output **Out** is connected to C.

ULN2003B
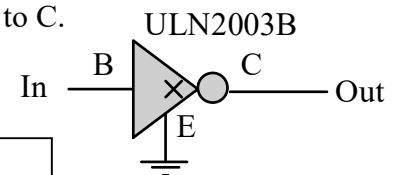
What is **Out** if **In** is 3.3V? ....

What is **Out** if **In** is 0.0V? ....

(12) Question 3. There is a 16-bit signed global variable called **Stuff**.
```
      AREA   DATA,ALIGN=2
Stuff SPACE 2
```
Write Cortex M assembly subroutine that performs the same operation as this C function

```
int16_t Stuff;
void ShiftandAdd(void){
  Stuff = (Stuff>>2)+5;
}
```

(5) Question 4. There is a 32-bit constant called **Thing**.
```
      AREA     |.text|, CODE, READONLY, ALIGN=2
Thing DCD 0xF012349A
```
What is the value of R0 in hex after this assembly code is executed?
```
  LDR   R1,=Thing
  LDRSB R0,[R1]
```

**(12) Question 5.** Show the declaration of a C function that finds the minimum value of an array. The length of the array is fixed at 1000. A pointer to the array is passed by reference into the function. The function returns the smallest value in the array. The function prototype is
```
int8_t min(int8_t data[1000]);
```

**(10) Question 6.** Interface eight switches to Port B using positive logic.

**(5) Part a)** For full credit, design the hardware/software interface that uses the fewest number of external components.
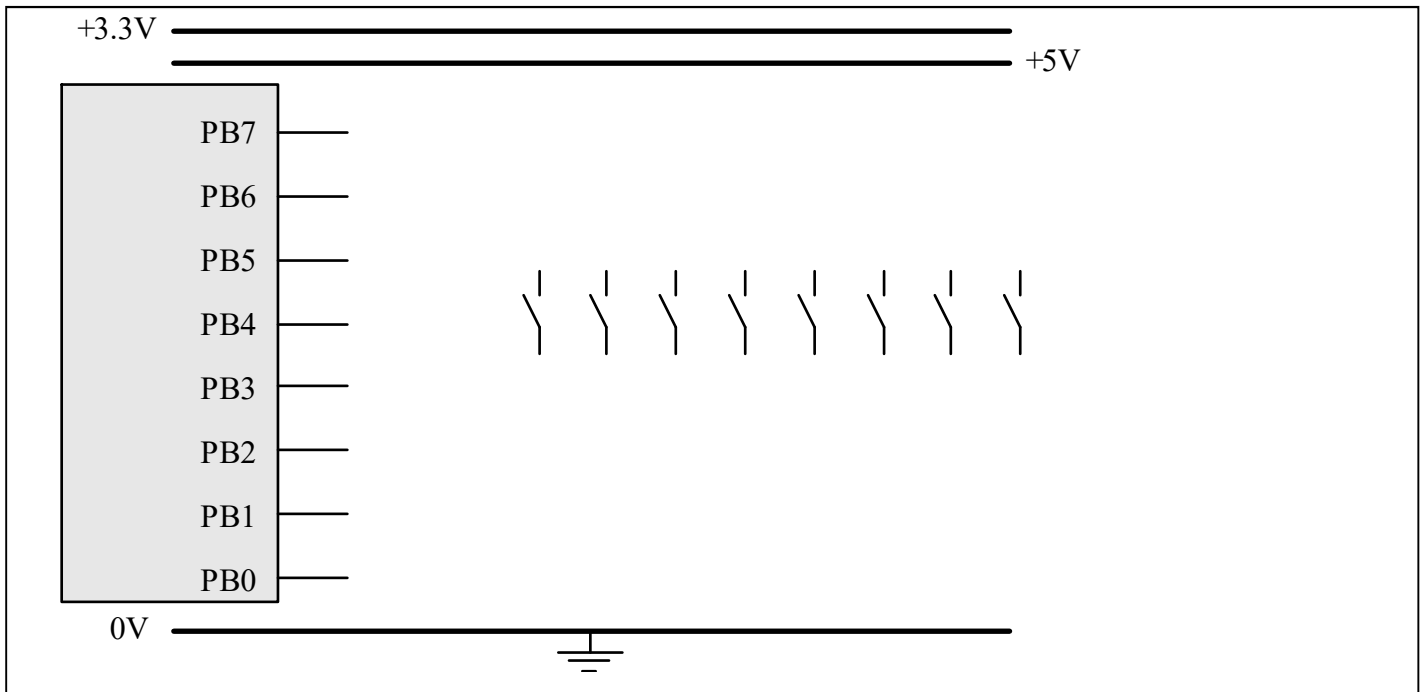


**(5) Part b)** During initialization what values should you write into the following registers? For registers you would not initialize, enter NA into the box. You may assume the Port B clock has been activated.

GPIO_PORTB_DATA_R ....................................

GPIO_PORTB_DIR_R ....................................

GPIO_PORTB_PUR_R ....................................

GPIO_PORTB_PDR_R ....................................

GPIO_PORTB_DEN_R ....................................

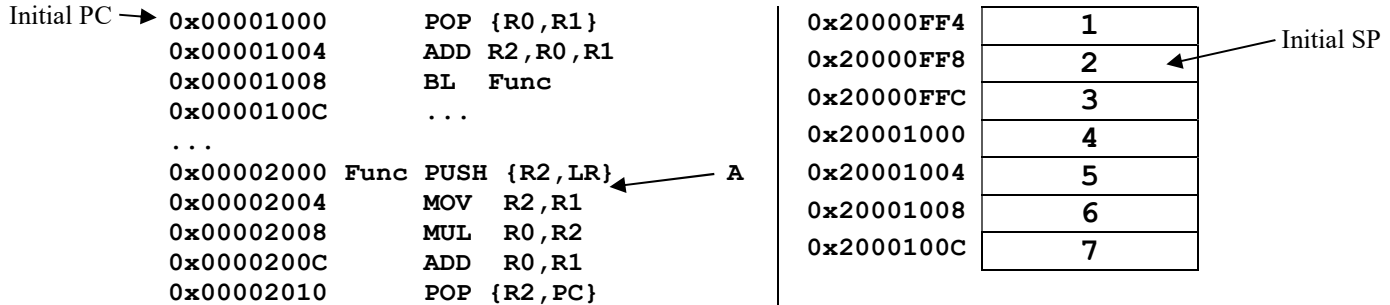**(10) Question 7.** Interface an LED to PA7 using positive logic.
**(5) Part a)** The desired LED operating point is 2V, 1mA. The microcontroller output high voltage is 3.2V, the microcontroller output low voltage is 0.1V. The ULN2003B output low voltage is 0.3V. For full credit, design an interface that uses the fewest number of external components. For any resistor(s) you use, show your work for determining the resistor value(s).
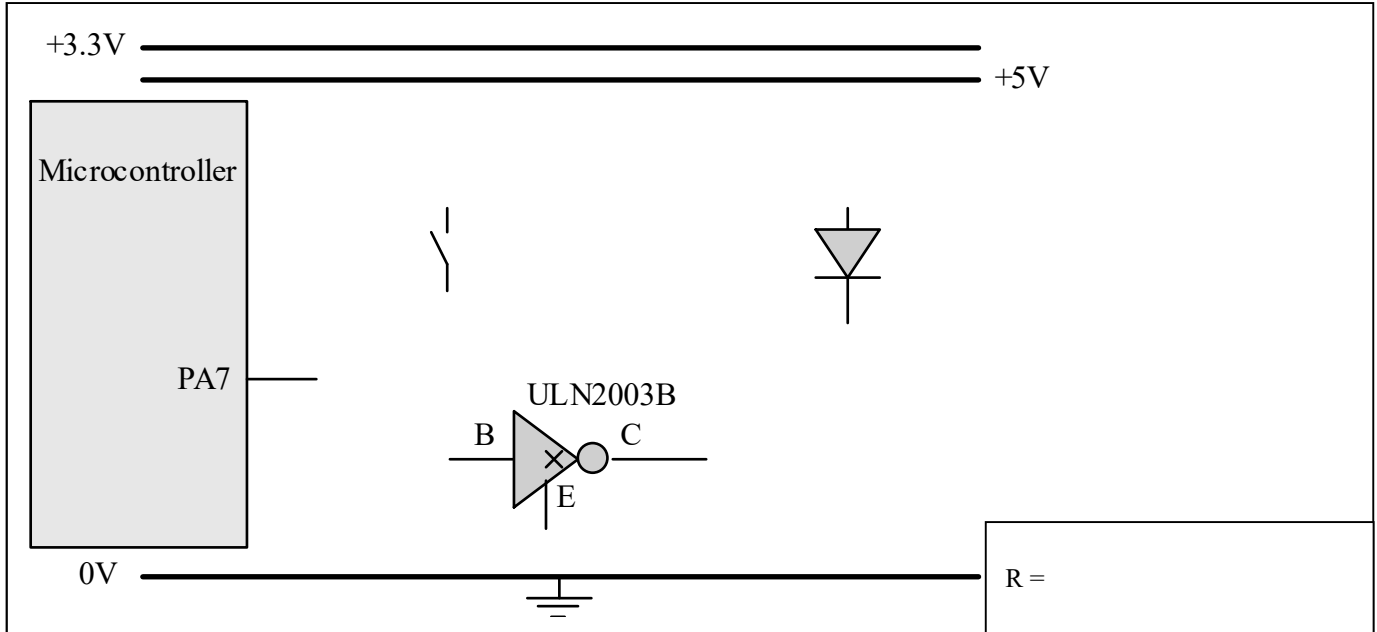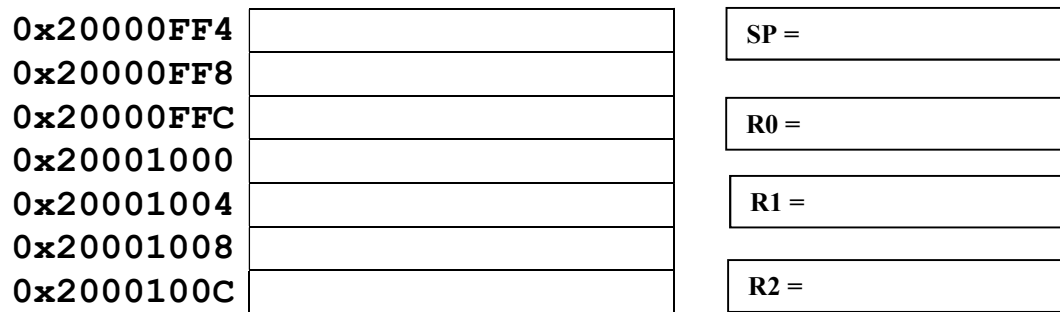


**(5) Part b)** Assume Port A has been initialized so PA7 is an output. Write a function in C that accepts an input parameter (0 or 0x80) and writes to Port A in a friendly manner. Include both the prototype and the declaration of the function.

```
#define GPIO_PORTA_DATA_R (*((volatile uint32_t *)0x400043FC))
```

**(8) Question 8.** Assume the value of the Stack pointer (SP) is **0x20000FF8** when the following code sequence starts execution (i.e., **PC=0x00001000**). The initial memory contents in and around the SP are given on the right. When drawing the stack contents, you need only to show values on the stack that represent actual valid stack data.

Initial PC →
```
0x00001000   POP {R0,R1}
0x00001004   ADD R2,R0,R1
0x00001008   BL   Func
0x0000100C   ...
...
0x00002000 Func PUSH {R2,LR}   ─── A
0x00002004   MOV R2,R1
0x00002008   MUL R0,R2
0x0000200C   ADD R0,R1
0x00002010   POP {R2,PC}
```

| Address | Value |
|---|---|
| 0x20000FF4 | 1 |
| 0x20000FF8 | 2 ← Initial SP |
| 0x20000FFC | 3 |
| 0x20001000 | 4 |
| 0x20001004 | 5 |
| 0x20001008 | 6 |
| 0x2000100C | 7 |

Staring with PC=0x00001000 execute until arrow A. Give the SP value and stack contents after executing of the **PUSH** instruction, as shown by arrow A. Also give the values stored in R0, R1, and R2.

| Address | |
|---|---|
| **0x20000FF4** | |
| **0x20000FF8** | |
| **0x20000FFC** | |
| **0x20001000** | |
| **0x20001004** | |
| **0x20001008** | |
| **0x2000100C** | |

SP =

R0 =

R1 =

R2 =

**(25) Question 9.** In this question there are two microcontrollers, such that the two Port B's are connected (PB7 to PB7, PB6 to PB6,…PB0 to PB0). The goal is to send a three-bit value from one microcontroller to the other.
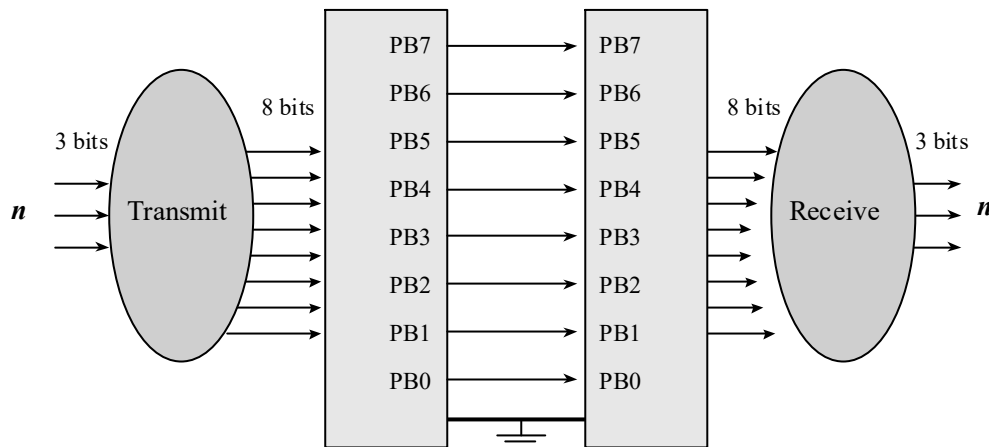


***Data flow graph***

The transmit software will be on the left microcontroller. At all times the transmitter must have exactly one of the Port B pins high. I.e., the 8-bit Port B data must be 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, or 0x80. You may assume the transmitter on the left has initialized all of Port B to be output, and the receiver on the right has initialized all of Port B to be input. On the transmitter, design an assembly function that accepts a 3-bit value in R0 (let $n$ be the value in R0, you may assume $0 \le n \le 7$) and writes to Port B the value $2^n$. On the receiver design an assembly function that reads Port B (knowing the Port B value will be restricted to $2^n$ for some $0 \le n \le 7$) and returns in R0 the value $n$.

```
GPIO_PORTB_DATA_R EQU 0x400053FC
```

```
;Input: R0=n is 0 to 7
;Set PortB data to 2^n
Transmit
```

```
;Read PortB data, which will be 2^n
;Output: R0=n is 0 to 7
Receive
```

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
```
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, return address in LR
BLX  Rm      ; branch to subroutine indirect specified by Rm
```
Interrupt instructions
```
CPSIE  I                ; enable interrupts  (I=0)
CPSID  I                ; disable interrupts (I=1)
```

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)
```
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
```
Notes Ra Rd Rm Rn Rt represent 32-bit registers
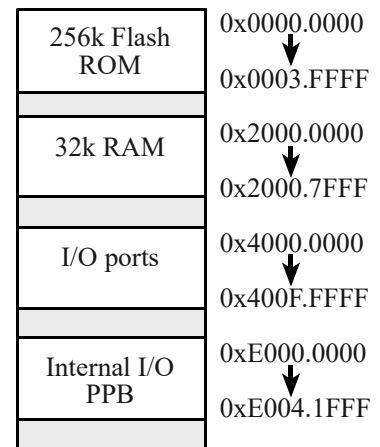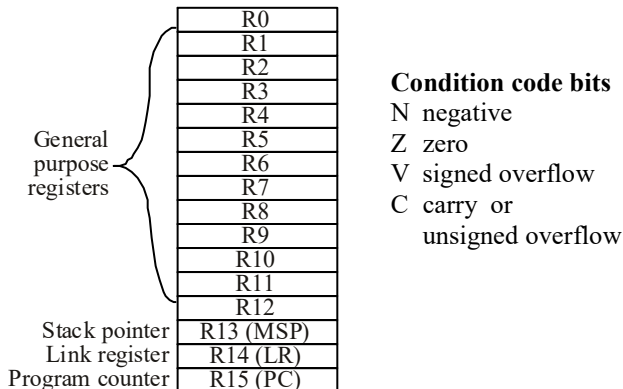```
    value   any 32-bit value: signed, unsigned, or address
    {S}   if S is present, instruction will set condition codes
    #im12   any value from 0 to 4095
    #im16   any value from 0 to 65535
    {Rd,}   if Rd is present Rd is destination, otherwise Rn
    #n   any value from 0 to 31
    #off   any value from -255 to 4095
    label   any address within the ROM of the microcontroller
    op2   the value generated by <op2>
```
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, where X and Y are hexadecimal digits:
```
- produced by shifting an 8-bit unsigned value left by any number of bits
- in the form **0x00XY00XY**
- in the form **0xXY00XY00**
- in the form **0xXYXYXYXY**

| General purpose registers | R0 |
| | R1 |
| | R2 |
| | R3 |
| | R4 |
| | R5 |
| | R6 |
| | R7 |
| | R8 |
| | R9 |
| | R10 |
| | R11 |
| | R12 |
| Stack pointer | R13 (MSP) |
| Link register | R14 (LR) |
| Program counter | R15 (PC) |

**Condition code bits**
N  negative
Z  zero
V  signed overflow
C  carry  or
   unsigned overflow

| | |
|---|---|
| 256k Flash ROM | 0x0000.0000 ↓ 0x0003.FFFF |
| 32k RAM | 0x2000.0000 ↓ 0x2000.7FFF |
| I/O ports | 0x4000.0000 ↓ 0x400F.FFFF |
| Internal I/O PPB | 0xE000.0000 ↓ 0xE004.1FFF |

```
    DCB  1,2,3 ; allocates three 8-bit byte(s)
    DCW  1,2,3 ; allocates three 16-bit halfwords
    DCD  1,2,3 ; allocates three 32-bit words
    SPACE 4    ; reserves 4 bytes
```