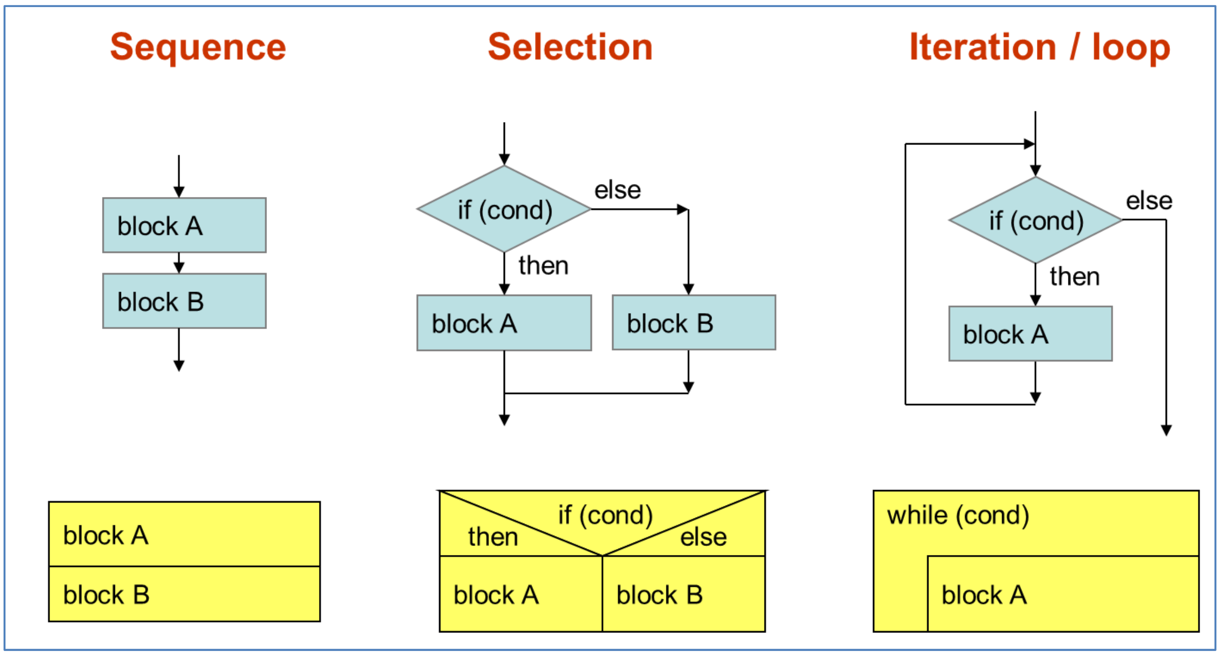
CT1 Exercises for Control Structures

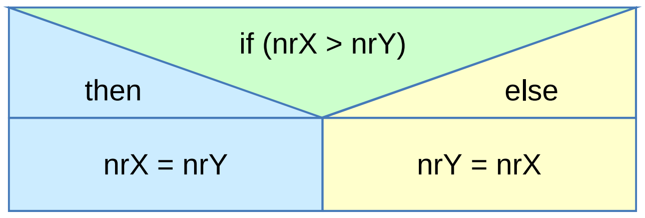
Content



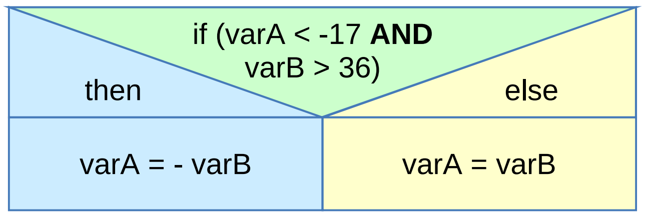
Exercise 1 – Selection/Branch

Encode the following Structograms into Flowchart, C- and ARM Assembly-language

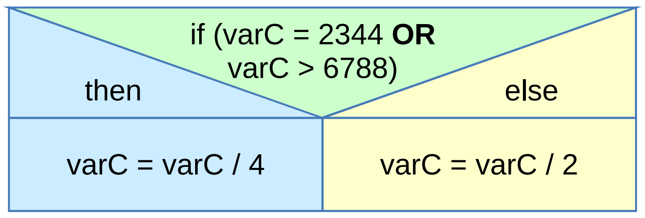
A) If-Then-Else with **unsigned** 8-bit variables



B) If-Then-Else with **signed** 8-bit variables



C) If-Then-Else with **signed** 16-bit variables



Exercise 2 – For-Loops

A) Write a for-loop in C- and ARM Assembly-language.

B) Compare your Assembly-language implementation with the compiler generated one.

   Hint: In the Keil uVision5 IDE

   1) create an empty C-language project (according to the respective introduction documents)
   2) add the C-language for-loop to the empty main function
   3) compile the project
   4) set a breakpoint in at the first line of the main function
   5) start debugging the program and let it run into the breakpoint
   6) compare your Assembly-language implementation of the for-loop with the compiler generated one

   Hint: for the purpose of this exercise, define your variables global and as “volatile” – this tells the compiler to not optimize away the access to the variables since they are not used otherwise.

Exercise 3 – From Code to Structogram

A) Analyze the following Assembly-language code and derive from this the matching structogram.

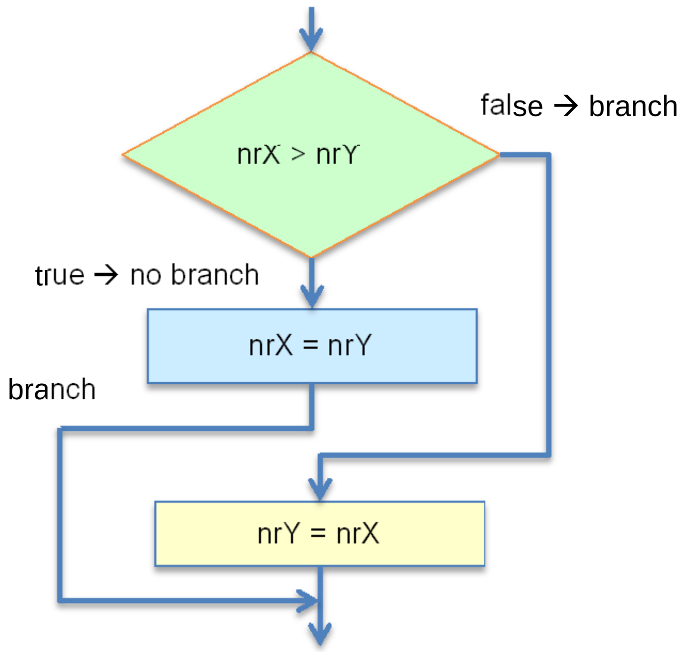
B) What result is stored in “outstr”?

```
        AREA progCode, CODE, READONLY
        THUMB

main    PROC
        EXPORT main

        LDR    R0,=srcstr
        LDR    R1,=outstr
        MOVS   R2,#0
cond    LDRB   R3,[R0,R2]
        CMP    R3,#0
        BEQ    endloop
        CMP    R3,#60
        BLO    store
        CMP    R3,#90
        BHI    store
        ADDS   R3,R3,#32
store   STRB   R3,[R1,R2]
        ADDS   R2,R2,#1
        B      cond
endloop STRB   R3,[R1,R2]

endless B      endless
        ENDP
srcstr  DCB    "This IS mY TestStriNG", 0

        AREA progData, DATA, READWRITE
outstr  SPACE 50

        END
```

Solutions

Exercise 1:

A) If-Then-Else with **unsigned** 8-bit variables



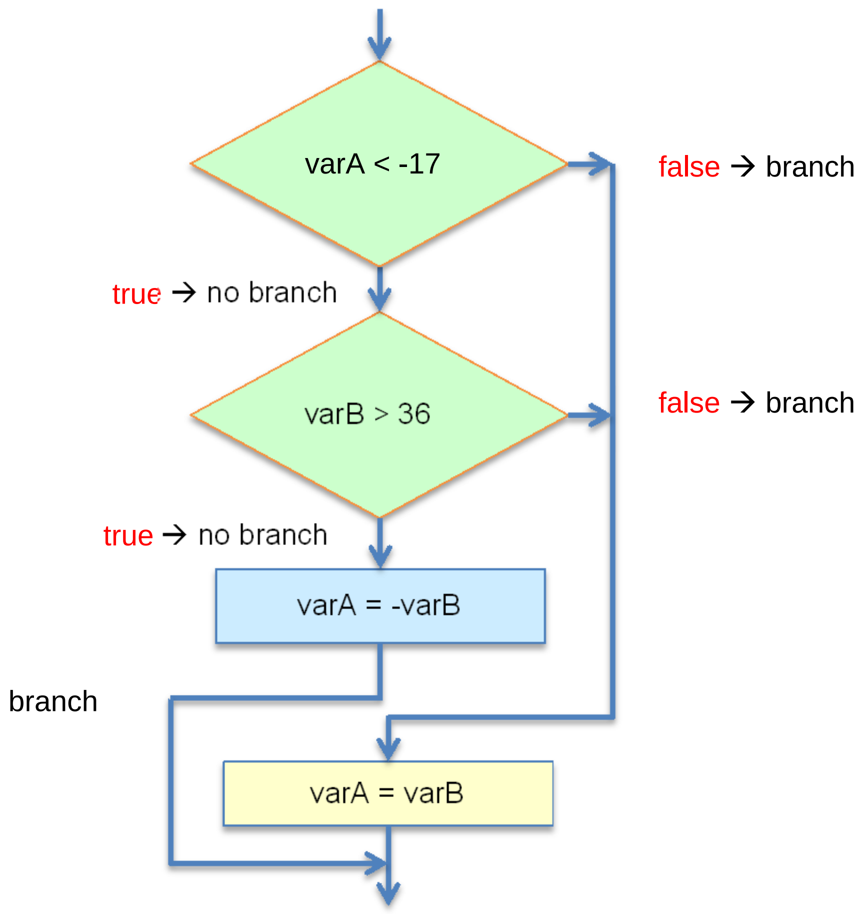
C

```
uint8_t nrX = ...;
uint8_t nrY = ...;
if (nrX > nrY) {
    nrX = nrY;
} else {
    nrY = nrX;
}
```

Assembly

```
        AREA progCode, CODE, READONLY
        THUMB

main    PROC
        EXPORT main

        LDR   R6,=nrX ; R6 = address of nrX
        LDRB  R0,[R6] ; R0 = byte stored at nrX
        LDR   R7,=nrY ; R7 = address of nrY
        LDRB  R1,[R7] ; R1 = byte stored at nrY
        CMP   R0, R1
        BLS   else      ; *unsigned* comparison
        STRB  R1,[R6] ; store nrY value at nrX
        B     endif
else    STRB  R0,[R7] ; store nrX value at nrY
endif

endless B      endless
        ENDP

        AREA progData, DATA, READWRITE

nrX     DCB      0x01 ; some 8 bit value
nrY     DCB      0xFF ; some other 8 bit value

        END
```

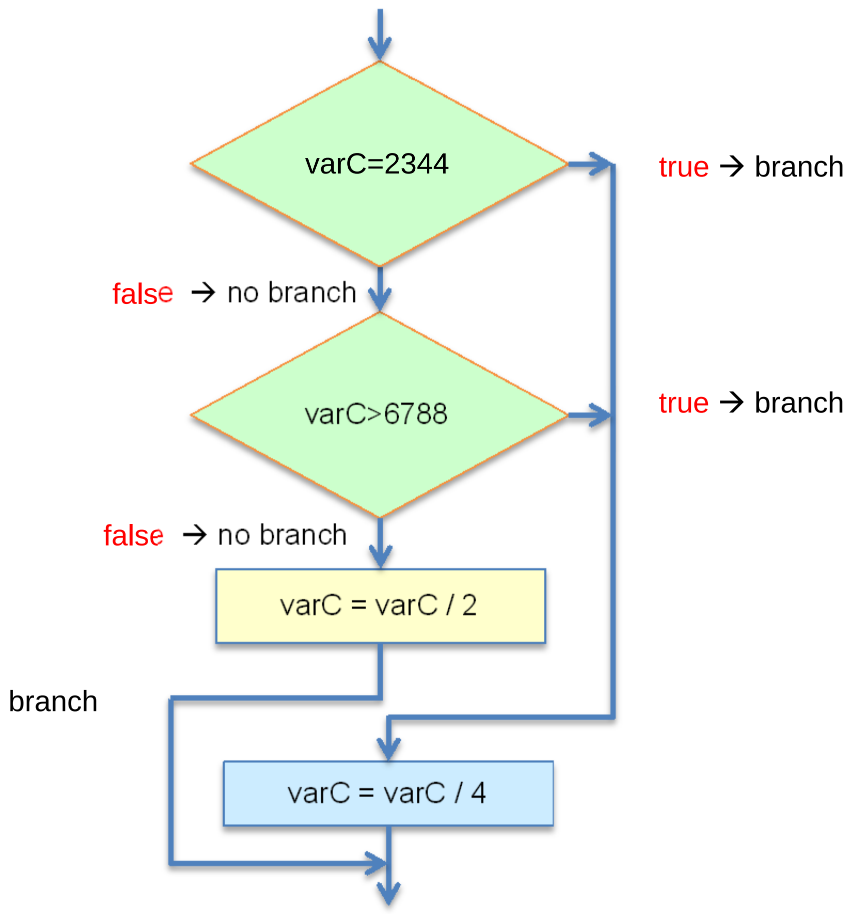
B) If-Then-Else with **signed** 8-bit variables



C	Assembly

```
int8_t varA = ...;
int8_t varB = ...;
if (varA < -17 && varB > 36) {
    varA = -varB;
} else {
    varA = varB;
}
```
	```
	        AREA progCode, CODE, READONLY
	        THUMB
	main    PROC
	        EXPORT  main

	        LDR   R6,=varA ; R6=address of varA
	        LDRB  R0,[R6]  ; R0=byte stored at varA
	        SXTB  R0,R0    ; extend signed varA
	        LDR   R7,=varB ; R7=address of varB
	        LDRB  R1,[R7]  ; R1=byte stored at varB
	        SXTB  R1,R1    ; extend signed varB
	        MOVS  R2,#17   ; +17
	        RSBS  R2,R2    ; -17
	        CMP   R0,R2
	        BGE   else     ; *signed* comparison
	        CMP   R1,#36
	        BLE   else     ; *signed* comparison
	        RSBS  R1,R1,#0 ; R1=-R1
	        STRB  R1,[R6]  ; varA = -varB
	        B     endif
	else    STRB  R1,[R6]  ; varA = varB
	endif
	endless B     endless
	        ENDP

	        AREA progData, DATA, READWRITE
	varA    DCB    123 ; some 8 bit value
	varB    DCB     45 ; some other 8 bit value
	        END
	```

C) If-Then-Else with **signed** 16-bit variables



C	Assembly

```
int16_t varC = ...;
if (varC == 2344 || varC > 6788){
    varC = varC / 4;
} else {
    varC = varC / 2;
}
```
	```
	        AREA progCode, CODE, READONLY
	        THUMB
	main    PROC
	        EXPORT main
	        LDR   R7,=varC ; R7=address of varC
	        LDRH  R0,[R7]  ; R0=value stored at varC
	        SXTH  R0,R0    ; extend signed varC
	        LDR   R2,=2344
	        CMP   R0,R2
	        BEQ   then     ; *signed* comparison
	        LDR   R2,=6788
	        CMP   R0,R2
	        BGT   then     ; *signed* comparison
	        ASRS  R0,R0,#1 ; R0 = R0 / 2
	        STRH  R0,[R7]  ; varC = varC / 2
	        B     endif
	then    ASRS R0,R0,#2  ; R0 = R0 / 4
	        STRH R0,[R7]   ; varC = varC / 4
	endif
	endless B      endless
	        ENDP

	        AREA progData, DATA, READWRITE
	varC    DCW    1234 ; some 16 bit value
	        END
	```

Exercise 2:

A) For-loop

C

```c
#include <utils_ctboard.h>
#include <stdint.h>
...
int32_t = 0;
int32_t count = 0;
for(i = 0; i < 10; i++) {
    count++;
}
```

Assembly

```
        AREA progCode, CODE, READONLY
        THUMB
main    PROC
        EXPORT main

        LDR   R6,=i      ; R6=address of i
        LDR   R0,[R6]    ; R0=value at i
        LDR   R7,=count  ; R7=address of count
        LDR   R1,[R7]    ; R1=value at count
        B     cond
loop    ADDS  R0,R0,#1
        ADDS  R1,R1,#1
cond    CMP   R0, #10
        BLT   loop        ; *signed* comparison
        STR   R0,[R6]     ; store final i
        STR   R1,[R7]     ; store final count
endless B      endless
        ENDP

        AREA progData, DATA, READWRITE
i       DCD    0
count   DCD    0
        END
```

B) Compare hand-crafted Assembly version to generated Assembly version

C

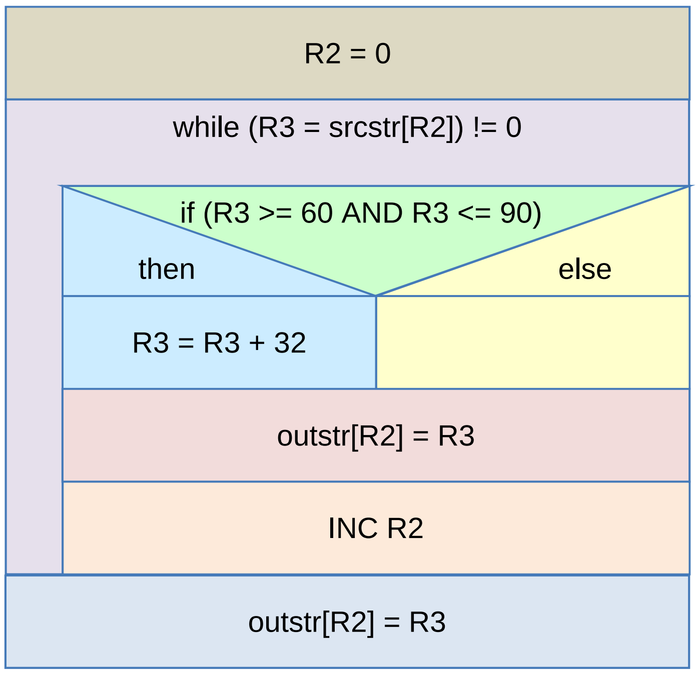
```c
#include <utils_ctboard.h>
#include <stdint.h>

int32_t i = 0;
int32_t count = 0;
int main()
{
    for(i = 0; i < 10; i++) {
        count++;
    }
}
```

Generated Assembly (aggressively optimized: -O3)

```
...
0x08000254 4905 LDR  r1,[pc,#20] ; @0x0800026C
0x08000256 2000 MOVS r0,#0x00
0x08000258 6008 STR  r0,[r1,#0x00]
0x0800025A 684A LDR  r2,[r1,#0x04]
0x0800025C 1C40 ADDS r0,r0,#1
0x0800025E 1C52 ADDS r2,r2,#1
0x08000260 280A CMP  r0,#0x0A
0x08000262 DBFB BLT  0x0800025C
0x08000264 C105 STM  r1!,{r0,r2}
...
0x0800026C 0000 DCW  0x0000
0x0800026E 2000 DCW  0x2000
```

Exercise 3:

A) The structorgram is



B) The resulting text is a null terminated string of all caps from the original string:

```
THIS IS MY TESTSTRING
```