SCHEDA **ASE1803**	Data: **14 Febbraio 2018**
Cognome	Nome

ESERCIZIO N°1
8 punti

Realizzare un sottoprogramma per il microcontrollore AVR XMEGA256A3BU, in grado di estrarre da una stringa - posta in memoria a partire dall'indirizzo contenuto in X - una stringa di *n* caratteri a partire dalla posizione *m*. Se, a partire dalla posizione assegnata, non sono presenti *n* caratteri, la stringa risultante si limiterà ai caratteri disponibili. Il risultato dovrà essere posto in memoria a partire dall'indirizzo contenuto in Y. Una stringa, la cui massima lunghezza può essere di 255 caratteri (codici ASCII su 8 bit), è composta da un primo byte che contiene l'indicazione della lunghezza della stringa stessa (nel caso di stringa vuota l'indicazione di lunghezza è 0) seguito dai codici dei caratteri che compongono la stringa. I valori di *n* e *m* sono contenuti in R16 e R17 rispettivamente; i comando con $m = 0$ e $m = 1$ si equivalgono.

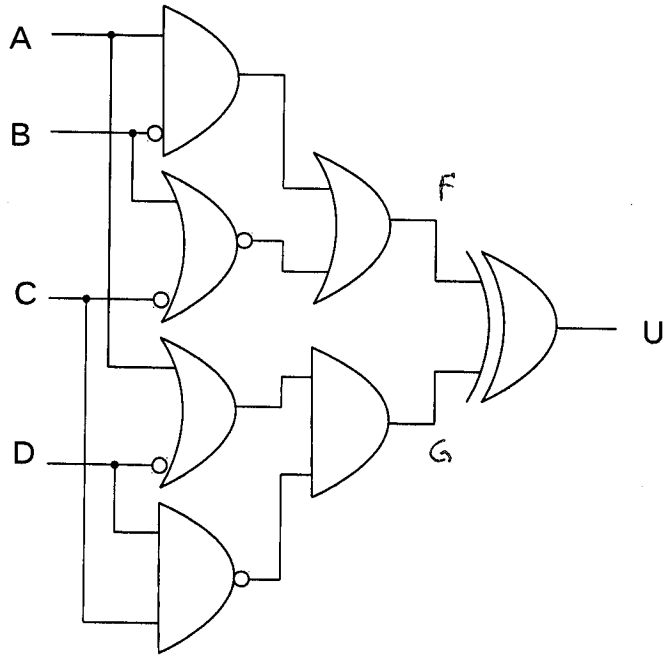
ESERCIZIO N°2
4 punti

Codificare (facendo eventualmente ricorso al data sheet) in valore binario ed esadecimale l'insieme delle tre seguenti istruzioni assembly di un microcontrollore XMEGA. Individuare quale condizione deve soddisfare il valore iniziale di R20 affinché il segmento di codice non resti bloccato permanentemente nel loop.

```
loop: ANDI R20,133
      SBRC R20,7
      BRNE loop
```

ESERCIZIO N°3
5 punti

Realizzare in forma NAND-NAND ottima la seguente rete combinatoria.



ESERCIZIO N°4

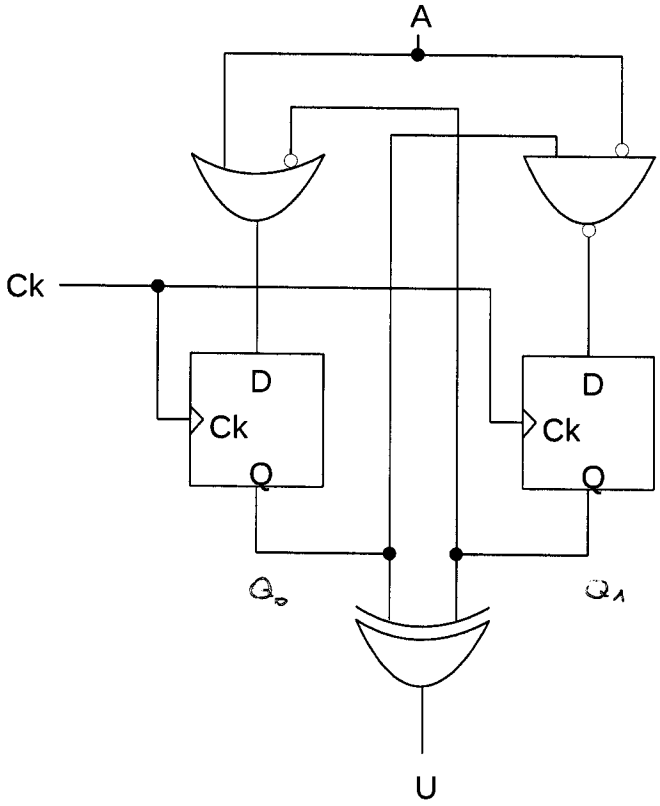
5 punti

Realizzare, se possibile con un approccio modulare, una rete combinatoria a 6 ingressi (2 numeri binari di 3 bit ciascuno) in grado di indicare con un valore alto in uscita quando uno dei due ingressi è maggiore o uguale all'altro.

ESERCIZIO N°5

5 punti

Determinare tipologia architetturale e grafo di flusso della seguente macchina sequenziale sincrona.



ESERCIZIO N°6

6 punti

Progettare una rete combinatoria, facente uso di blocchi noti (porte logiche elementari, multiplexer, full-adder, ecc.) in grado di eseguire la somma di 3 numeri relativi a 8 bit, rappresentati rispettivamente come MS, Traslazione e C1 e il cui risultato (per il risultato scegliere il numero di bit minimo che garantisce in ogni caso la rappresentabilità) sia rappresentato in C2.

1

Realizzare un sottoprogramma per il microcontrollore AVR XMEGA256A3BU, in grado di estrarre da una stringa - posta in memoria a partire dall'indirizzo contenuto in X - una stringa di *n* caratteri a partire dalla posizione *m*. Se, a partire dalla posizione assegnata, non sono presenti *n* caratteri, la stringa risultante si limiterà ai caratteri disponibili. Il risultato dovrà essere posto in memoria a partire dall'indirizzo contenuto in Y. Una stringa, la cui massima lunghezza può essere di 255 caratteri (codici ASCII su 8 bit), è composta da un primo byte che contiene l'indicazione della lunghezza della stringa stessa (nel caso di stringa vuota l'indicazione di lunghezza è 0) seguito dai codici dei caratteri che compongono la stringa. I valori di *n* e *m* sono contenuti in R16 e R17 rispettivamente; i comando con $m = 0$ e $m = 1$ si equivalgono.

```
extract:
 push R16 //n numero di caratteri della sottostringa
 push R17 //m posizione iniziale
 push R18 //Ltot
 push R19 //ausiliario
 push XL //puntatore a stringa1
 push XH
 push YL //puntatore al risultato
 push YH
 clr R1 //registro nullo
 tst R16
 breq nulstring //stringa estratta vuota
 tst R17
 breq zero
 dec R17 //se m<>0 esegue m-1 (0 parte dal primo carattere, 1 dal secondo, ecc.)
zero:
 ld R18,X+ //lunghezza della stringa di partenza
 sub R18,R17 //calcola i caratteri disponibili da estrarre
 brcs nulstring //nessun carattere disponibile
 breq nulstring //nessun carattere disponibile
 cp R18,R17
 brcc forward //ci sono abbastanza caratteri disponibili
 mov R16,R18 //lunghezza effettiva della stringa estratta
forward:
 st Y+,R16 //scrive la lunghezza di stringa
 add XL,R17
 adc XH,R1 //sistema il puntatore a partire dai caratteri interessanti
loop:
 ld R19,X+
 st Y+,R19
 dec R16
 brne loop //scrive i caratteri uno alla volta
 rjmp restore
nulstring:
 st Y,R1 //scrive solo la lunghezza nulla
restore:
 pop YH
 pop YL
 pop XH
 pop XL
 pop R19
 pop R18
 pop R17
 pop R16
 ret
```

(2)

Codifica le istruzioni

coop: ANDI R20, 133

OPCODE:
0111 KKKK dddd KKKK

20 = 0b10100 (d)
133 = 0b10000101 (k)

quindi { 0111 1000 0100 0101 bin
7 8 4 5 hex

SBRC R20, 7

OPCODE:
1111 110r rrrr 0bbb

20 = 0b10100 (r)
7 = 0b111 (b)

quindi { 1111 1101 0100 0111 bin
F D 4 7 hex

BRNE coop (-3)

OPCODE
1111 01KK KKKK K001

-3 = 0b1111101

quindi { 1111 0111 1110 1001 bin
F 7 E 9 hex

Perché il programma non si blocchi, deve eseguire la skip, quindi il MSB di R20 deve essere NULLO.

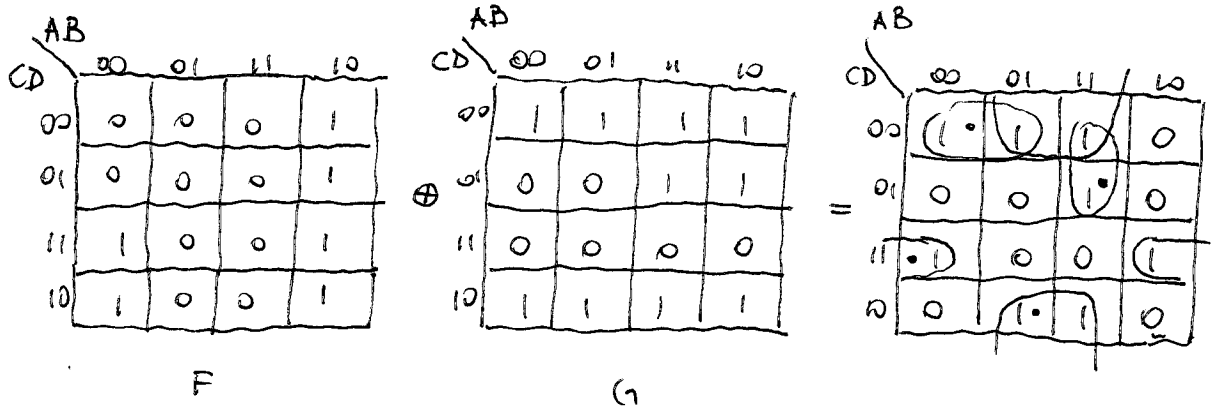
(3) Sintesi NAND - NAND

(a partire dalla SP e si modifica secondo il teorema di DeMorgan)

Esprimo la funzione in termini booleani

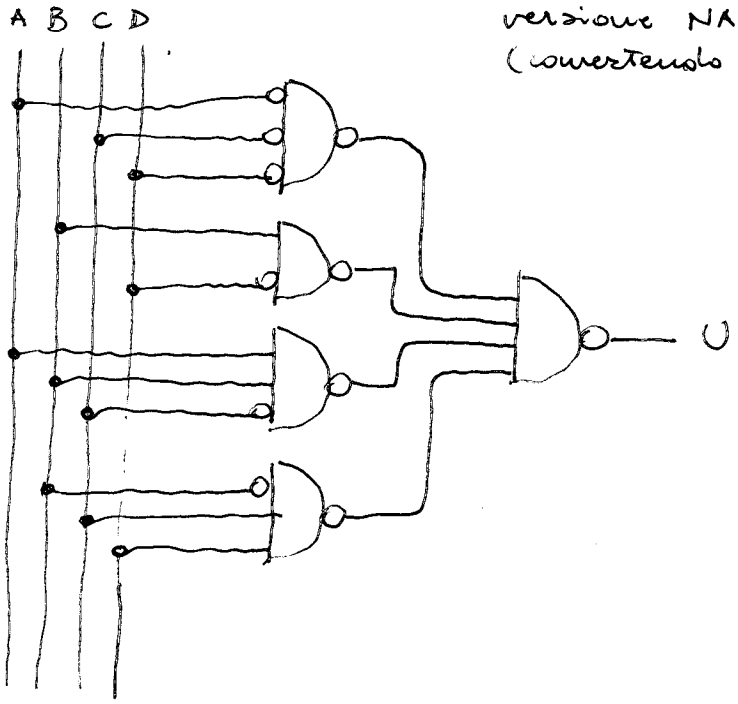
$F = A\bar{B} + C\bar{B}$ { DeMorgan alla NOR }

$G = (A + \bar{D})(\bar{C} + \bar{D})$ { ... alla NAND }



Sintesi SP (tutti ESSENZIALI)

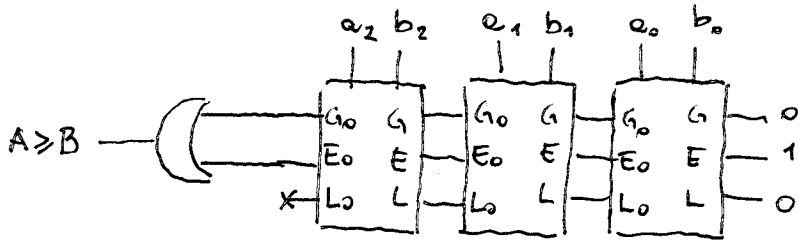
$U = \bar{A}\bar{C}\bar{D} + B\bar{D} + AB\bar{C} + \bar{B}CD$



versione NAND - NAND
(convertendo la OR finale)

④ Comparatore ≥

Considero la versione "LSB first"

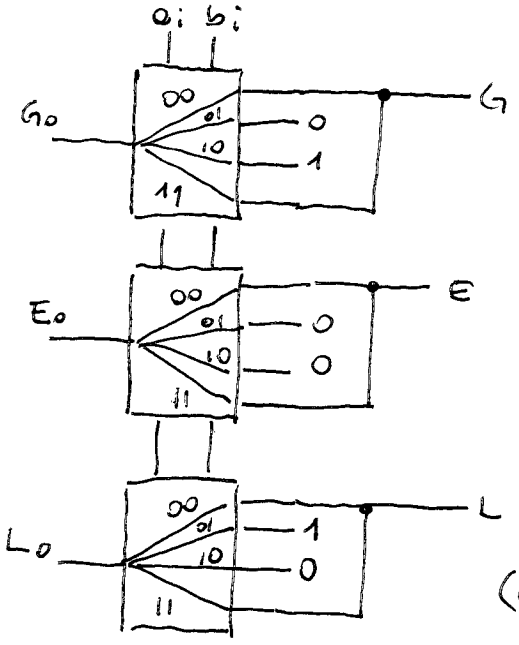


Funzione del comparatore di bit

a_i	b_i	G	E	L	G_0	E_0	L_0
0	1	–	–	–	0	0	1
1	0	–	–	–	1	0	0
0	0	0	1	0	0	1	0
0	0	1	0	0	1	0	0
0	0	0	0	1	0	0	1
1	1	0	1	0	0	1	0
1	1	1	0	0	1	0	0
1	1	0	0	1	0	0	1

casi significativi

Sintesi con multiplexer



(questo si può anche omettere, vista la richiesta del testo)

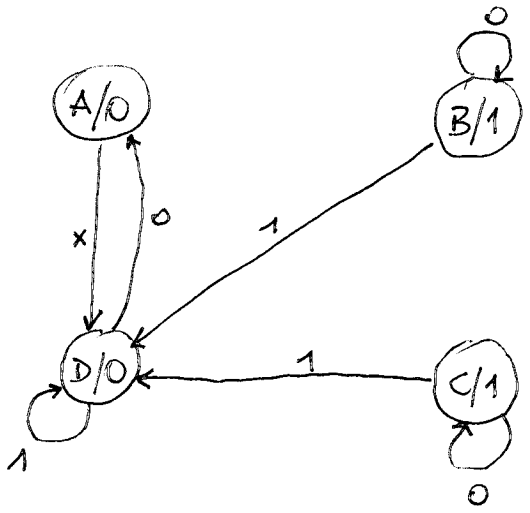
(5) La macchina è sincrona (D-FF non trasp. con stesso Ck)
La macchina è di Moore (l'uscita dipende SOLO dallo stato)

Tabella di transizione

$A+\bar{Q}_0$; $A+\bar{Q}_1$

SP	Q_1	Q_0	A	Q_1^+	Q_0^+	U	SF
A	0	0	0	1	1	0	D
			1	1	1		D
B	0	1	0	0	1	1	B
			1	1	1		D
C	1	0	0	1	0	1	C
			1	1	1		D
D	1	1	0	0	0	0	A
			1	1	1		D

GRAFO

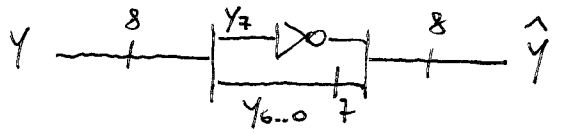


(6) Determinare il range della somma

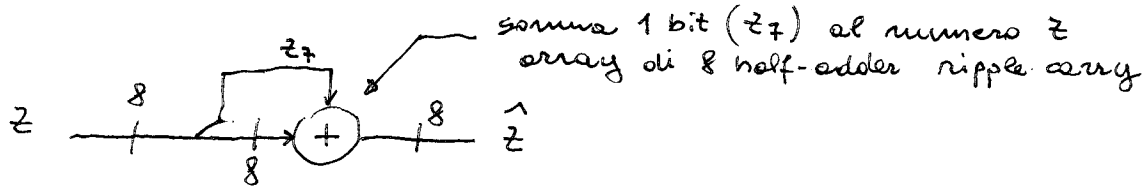
	MS(x)	T (y)	C1(z)	Risultato	
Max positivo	127	127	127	+381	} Servono 10b (in C2)
min negativo	-127	-128	-127	-382	

Strategia: converto tutti i formati in C2, estendo a 10b ed eseguo la somma. Nelle conversioni non si può avere OF

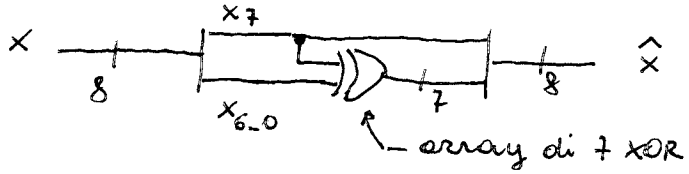
1) conversione T-C2



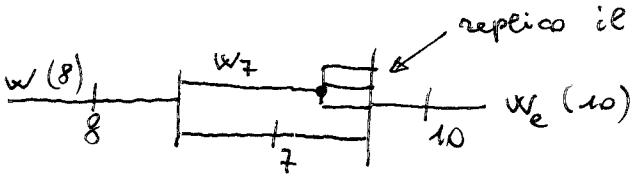
2) conversione C1-C2



3) conversione MS-C1 (per passare in C2 uso il blocco precedente)



4) estensione del segno



5) somma finale