COMPITO di ARCHITETTURA DEI CALCOLATORI del 14-02-2018 (rev1.1) **MATRICOLA**_____

DA RESTITUIRE INSIEME AGLI ELABORATI e A TUTTI I FOGLI
→ NON USARE FOGLI NON TIMBRATI
→ ANDARE IN BAGNO PRIMA DELL'INIZIO DELLA PROVA
→ NO FOGLI PERSONALI, NO TELEFONI, SMARTPHONE, ETC

COGNOME_____

NOME_____

SVOLGIMENTO DELLA PROVA (selezionare una delle seguenti 4 opzioni):
☐ PER GLI STUDENTI DI "ARCHITETTURA DEI CALCOLATORI – A.A. 2015/16 e 16/17": es. N.1+2+3+7.

NOTA: per l'esercizio 7 dovranno essere consegnati DUE files: il file del programma VERILOG e il file relativo all'output (screenshot o copy/paste)

1) [18/38] Trovare il codice assembly MIPS corrispondente al seguente programma **(usando solo e unicamente istruzioni della tabella sottostante e rispettando le convenzioni di utilizzazione dei registri dell'assembly MIPS** riportate qua sotto per riferimento).

```
float sqrt1(float n){
    float low=0, high=n, xlow=high, xhigh=low
    float sqrt=0, res=0; int times = 1;
    if(n<0) return -1;
    while(n<1){ n=n*100; times++; }
    if(n<0) n=-1*n;
    while(res!=n){
        sqrt = (high+low)/2;
        res = sqrt*sqrt;
        if(res>n) high = sqrt;
        else if(res<n) low = sqrt;
        if(xlow==low && xhigh==high){
            break;
        }else{
            xlow = low;
            xhigh = high;
        }
    }
    for(int i=1; i<times; i++){
        sqrt=sqrt/10;
    }
    return sqrt;
}

int main() {
    float sq7,sq27;
    sq7=sqrt1(7);
    sq27=sqrt1(27);
    print_float(sq7);
    print_string("\n");
    print_float(sq27);
    print_string("\n");
    exit(0);
}
```

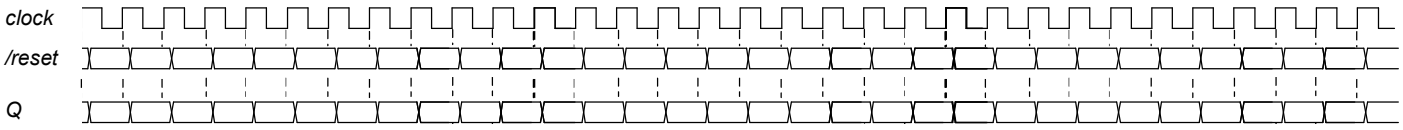
2) [8/38] Si consideri una cache di dimensione 128B e a 4 vie di tipo write-back/write-non-allocate. La dimensione del blocco e' 8 byte, il tempo di accesso alla cache e' 4 ns e la penalita' in caso di miss e' pari a 40 ns, la politica di rimpiazzamento e' LRU. Il processore effettua i seguenti accessi in cache, ad indirizzi al byte: 127, 113, 163, 111, 140, 161, 115, 224, 222, 241, 216, 313, 416, 523, 691, 716, 831, 910, 1011, 1118, 1231, 121. Tali accessi sono alternativamente letture e scritture. Per la sequenza data, ricavare il tempo medio di accesso alla cache, riportare i tag contenuti in cache al termine, i bit di modifica (se presenti) e la lista dei blocchi (ovvero il loro indirizzo) via via eliminati durante il rimpiazzamento ed inoltre in corrispondenza di quale riferimento il blocco e' eliminato.

3) [4/38] Assemblare il seguente programma MIPS, utilizzando la tabella sottostante e riportando il formato utilizzato e i valori in esadecimale di ciascun campo di quel formato (es. LW $2, 0($1) → FORMATO I: 23 1 2 0); il programma viene caricato all'indirizzo standard 0x0040'0000:

```
lab:    LW      $2, 0($1)
        SW      $4, 0($3)
        SLL     $5, $2, 10
        SRA     $6, $4, 12
        MULTU   $5, $6
        MFLO    $7
        JAL     fun
        SLT     $8, $7, $2
        BNE     $8, $0, lab
        JR      $31
fun:    DIV     $10, $12
        MFHI    $2
        JR      $31
```

4) Non assegnato
5) Non assegnato
6) Non assegnato

7) [8/38] **Realizzare** in Verilog (per studenti 2014 e anni precedenti --> v.nota finale) sia un serial carry counter a 4-bit che il relativo testbench: il clock ha un periodo di 10ns; il segnale _reset e' attivo basso: resta alto per 5ns, basso per 20ns, e poi ritorna alto per 600ns. Il contatore inizia il conteggio producendo sull'uscita Q il valore binario 0000 quando il segnale di /reset e'attivato, appena disattivato il /reset il conteggio prosegue. **Tracciare il diagramma di temporizzazione** come verifica della correttezza dell'unità riportando i segnali clock, /reset, uscita Q per la durata complessiva (625ns). Nota: si può svolgere l'esercizio su carta oppure con ausilio del simulatore salvando una copia dell'output (diagramma temporale) e del programma Verilog su USB-drive del docente. (Per studenti 2014 e anni precedenti descrivere il comportamento di questa rete e disegnare líntero diagramma di temporizzazione, come sopra specificato).



COMPITO di ARCHITETTURA DEI CALCOLATORI del 14-02-2018 (rev1.1) **MATRICOLA**_____

SOLUZIONE **COGNOME**_____

NOME_____

Instructions

Opcode+Funct (hexadecimal)	Instruction	Example	Meaning	Comments
00+20/00+21	**add**	`add/addu $1,$2,$3`	$1 = $2 + $3	(signed/unsigned) 3 operands; exception possible
00+22/00+23	**subtract**	`sub/subu $1,$2,$3`	$1 = $2 - $3	(signed/unsigned) 3 operands; exception possible
08/09	**add immediate**	`addi/addiu $1,$2,100`	$1 = $2 + 100	(signed/unsigned) + constant ; exception possible
00+18/00+19	**multiplication**	`mult/multu $1, $2`	Hi,Lo= $1 x $2	(signed/unsigned) 64-bit Product ; result in Hi,Lo
00+1A/00+1B	**division**	`div/divu $1, $2`	Hi= $1 % $2, Lo = $1 / $2	(signed/unsigned) division
00+10/00+12	**move from Hi / move from Lo**	`mfhi/mflo $1`	$1 = Hi ($1 = Lo)	Create copy of Hi (Create a copy of Lo)
00+2A/00+2B	**set on less than**	`slt/sltu $1,$2,$3`	if ($2 < $3) $1 = 1; else $1 = 0	(signed/unsigned) compare $2 and $3 (less than)
0A/0B	**set on less than immediate**	`slti/sltiu $1,$2,100`	if ($2 < 100) $1 = 1; else $1 = 0	(signed/unsigned) compare $2 and constant (less than)
00+24/25/26/27	**and / or / xor / nor**	`and/or/xor/nor $1,$2,$3`	$1=$2&$3 / $2|$3 / $2^$3 / !($2|$3)	3 register operands; Logical AND/OR/XOR/NOR
0C/0D/0E	**and /or / xor immediate**	`andi/ori/xori $1,$2,100`	$1 = $2 & 100 / $2 | 100 / $2 ^100	Logical AND/OR/XOR register, constant
00+00	**shift left logical**	`sll $1,$2,10`	$1 = $2 << 10	Shift left by constant
00+02/00+03	**shift right (l=logical,a=arithmetic)**	`srl/sra $1,$2,10`	$1 = $2 >> 10	Shift right by constant (for arithmetic: sign is preserved)
23/20	**load word / load byte**	`lw/lb $1,100($2)`	$1 = Memory[$2+100]	Data from memory to register
24	**load byte unsigned**	`lbu $1,100($2)`	$1 = Memory[$2+100]	Data from mem. To reg.; no sign extension
2B/28	**store word / store byte**	`sw/sb $1,100($2)`	Memory[$2+100] = $1	Data from register to memory
0F	**load upper immediate**	`lui $1,0x1234`	$1=0x1234'0000	load most significant 16 bits
PSEUDOINSTRUCTION	**load address**	`la $1,var`	$1 = &var	Load address of var (lui $1,H16(&var);ori $1, L16(&var)) H16/L16=high/low 16 bits of &var
02	**jump**	`j 10000`	go to 10000	Jump to target address
00+08	**jump register**	`jr $31`	go to $31	For switch, procedure return
03	**jump and link**	`jal 10000`	$31 = PC + 4;go to 10000	For procedure call
04	**branch on equal**	`beq $1,$2,100`	if ($1 = = $2) go to PC+4+100	Equal test; PC relative branch
05	**branch on not equal**	`bne $1,$2,100`	if ($1 != $2) go to PC+4+100	Not equal test; PC relative
00+0C	**syscall**	`syscall`	call OS service $v0	See table of system calls below
10+10,rs=10	**rfe**	`rfe`	shift right (k,e) bits in STATUS reg	Exit Kernel Mode, Enable Interrupts
PSEUDOINSTRUCTION	**branch unconditional**	`b 100`	go to PC+4+100	PC relative branch (e.g., beq $0,$0,100)
PSEUDOINSTRUCTION	**no operation**	`nop`	do nothing	Do nothing (e.g. sll $0,$0,0)
30	**load-linked**	`ll $1,100($2)`	$1=Memory[$2+100]	Read and start to monitor the given memory location
38	**store-conditional**	`sc $1,100($2)`	Memory[$2+100]=$1 or →	return 0 if a coherence action happens since the previous ll ($1 must be different from 0)
11+00 fmt=10/11	**add.s / add.d**	`add.x $f0,$f2,$f4`	$f0=$f2+$f4	Single and double precision add
11+01 fmt=10/11	**sub.s / sub.d**	`sub.x $f0,$f2,$f4`	$f0=$f2-$f4	Single and double precision subtraction
11+02 fmt=10/11	**mul.s / mul.d**	`mul.x $f0,$f2,$f4`	$f0=$f2*$f4	Single and double precision multiplication
11+03 fmt=10/11	**div.s / div.d**	`div.x $f0,$f2,$f4`	$f0=$f2/$f4	Single and double precision division
11+05 fmt=10/11	**abs.s / abs.d**	`abs.x $f0,$f2`	$f0=ABS($f2)	Single and double precision absolute value
11+06 fmt=10/11	**mov.s / mov.d**	`mov.x $f0,$f2`	$f0←$f2	Single and double precision move
11+07 fmt=10/11	**neg.s / neg.d**	`neg.x $f0,$f2`	$f0= – ($f2)	Single and double precision opposite value
11+3C(31,32,3D,3E,3F) fmt=10/11	**c.lt.s / c.lt.d (ne,eq,gt,le,ge)**	`c.lt.x $f0,$f2`	Temp=($f0<$f2)	Single and double: compare $f0 and $f2 <,=,!=,>,<=>=
11+00 fmt=4/0	**move to/from coprocessor 1**	`mtc1/mfc1 $1,$f2`	$f2=$1 / $1=$f2	Move $1 to/from C1reg. $f2 (no conversion)
10+00 fmt=4/0	**move to/from coprocessor 0**	`mtc0/mfc0 $1,$2`	$c2=$1 / $1=$c2	Move $1 to/from C0 reg. $f2 (no conversion)
11+00 fmt=6/2	**move to/from control reg of cop.1**	`ctc1/cfc1 $1,$cf2`	$cf2=$1 / $1=$cf2	Move $1 to/from C1-CONTROL register
11 fmt=8,ft=1/0	**branch on true/false**	`bc1t/bc1f label`	If (Temp = = true/false) go to label	Temp is 'Condition-Code'
31/39	**load/store floating point (32bit)**	`lwc1/swc1 $f0,0($1)`	$f0←Memory[$1] / Memory[$1]←$f0	Data from FP (C1) register to memory
11+21,fmt=10/11+22,fmt=11	**convert from/to single to/from double**	`cvt.d.s/cvt s.d $f0,$f2`	$f0=(double)$f2/$f0=(single)$f2	Type conversion
11+24,fmt=11/11+20	**convert from/to single to/from integer**	`cvt.w.s/cvt s.w $f1,$f0`	$f1=(int)$f0 / $f0=(single)$f2	Type conversion

Register Usage

Name	Reg. Num.	Usage
$zero	0	The constant value 0
$s0-$s7	16-23	Saved
$t0-$t9	8-15,24-25	Temporaires
$a0-$a3	4-7	Arguments

Name	Reg.Num.	Usage
$v0-$v1	2-3	Results
$fp, $sp	30,29	frame pointer, stack pointer
$ra, $gp	31,28	return address, global pointer
$k0-$k1	26,27	Kernel usage

Reg. Num.	Usage
$f0, $f2	Return values
$f12,$f14	Function arguments
$f20,$f22,$f24,$f26,$f28,$f30	Saved registers
$f4,$f6,$f8,$f10,$f16,$f18	Temporaries registers

System calls

Service Name	Service Num. ($v0)	INPUT Arguments	OUTPUT Arguments
print_int	1	$a0=integer to print	---
print_float	2	$f12=float to print	---
print_double	3	($f12,$f13)=double to print	---
print_string	4	$a0=address of ASCIIZ string to print	---
read_int	5	---	$v0=integer
read_float	6	---	$f0=float
read_double	7	---	$f0-f1=double
read_string	8	$a0=address of input buffer, $a1=max characters to read	---
sbrk	9	$a0=Number of bytes to be allocated	$v0=pointer to the allocated memory
exit	10	---	---

COMPITO di ARCHITETTURA DEI CALCOLATORI del 14-02-2018 (rev1.1) **MATRICOLA**_____

SOLUZIONE **COGNOME**_____

NOME_____

TRACCIA DELLA SOLUZIONE

ESERCIZIO 1

```
.data
    newline: .asciiz "\n"

.text
.globl main

main:
    addi $t0,$0,7     # t0=7
    mtc1 $t0,$f12     # f12=7
    cvt.s.w $f12,$f12 # f12=7.0
    jal sqrt1         # call function
    mov.s $f20,$f0    # save first result in f20

    addi $t0,$0,27    # t0=27
    mtc1 $t0,$f12     # f12=27
    cvt.s.w $f12,$f12 # f12=27.0
    jal sqrt1         # call function
    mov.s $f21,$f0    # save first result in f21

    mov.s $f12,$f20   # prepare arg for print
    addi $v0,$0,2     # print_float service #2
    syscall

    la $a0,newline    # print "\n"
    addi $v0,$0,4
    syscall

    mov.s $f12,$f21   # prepare arg for print
    addi $v0,$0,2     # print_float service #2
    syscall

    la $a0,newline    # print "\n"
    addi $v0,$0,4
    syscall
    addi $v0,$0,10    # exit
    syscall

sqrt1:
    # associate f4 -> low
    # associate f5 -> high
    # associate f6 -> xlow
    # associate f7 -> xhigh
    # associate f8 -> sqrt
    # associate f9 -> res
    # associate t9 -> times

    mtc1 $0,$f10      # f10=0
    cvt.s.w $f10,$f10 # f10=0.0
    mov.s $f4,$f10    # low=0.0
    mov.s $f8,$f10    # sqrt=0.0
    mov.s $f9,$f10    # res=0.0
    mov.s $f5,$f12    # high=n
    mov.s $f6,$f12    # xlow=high=n
    mov.s $f7,$f10    # xhigh=low=0.0
    addi $t9,$0,1     # times=1

    c.lt.s $f12,$f10  # if(n<0)
    bc1t retmeno1     # return -1

    addi $t0,$0,1     # t0=1
    mtc1 $t0,$f11     # f11=1
    cvt.s.w $f11,$f11 # f11=1.0

iniwhile1:
    c.lt.s $f12,$f11  # while(n<1)
    bc1f endwhile1    # skip while1

    addi $t0,$0,100   # t0=100
    mtc1 $t0,$f16     # f16=100
    cvt.s.w $f16,$f16 # f16=100.0
    mul.s $f12,$f12,$f16 # n=n*100

    addi $t9,$t9,1    # times++

    j iniwhile1

endwhile1:
    c.lt.s $f12,$f10  # if(n<0)
    bc1f notif1       # skip n=-1*n

    addi $t0,$0,-1    # t0=-1
    mtc1 $t0,$f16     # f16=-1
    cvt.s.w $f16,$f16 # f16=-1.0
    mul.s $f12,$f16,$f12  # n=-1*n

notif1:
iniwhile2:
    c.eq.s $f9,$f12   # check if res==n
    bc1t endwhile2    # if true skip while

    add.s $f10,$f5,$f6 # f10=(high+low)
    addi $t0,$0,2     # t0=2
    mtc1 $t0,$f16     # f16=2
    cvt.s.w $f16,$f16 # f16=2.0
    div.s $f8,$f10,$f16# sqrt=(high+low)/2

    mul.s $f9,$f8,$f8 # res=sqrt*sqrt

    c.le.s $f9,$f12   # if(res>n)
    bc1t notifw1      # skip high=sqrt

    mov.s $f5,$f8     # high=sqrt
    j notelseifw1     # skip eldeif condition

notifw1:
    c.lt.s $f9,$f12   # if(res<n)
    bc1f notelseifw1  # skip low=sqrt
    mov.s $f4,$f8     # low=sqrt

notelseifw1:
    c.eq.s $f6,$f4    # check if xlow==low
    bc1f elseifw2     # skip if

    c.eq.s $f7,$f5    # check if xhigh==high
    bc1f elseifw2     # skip if

    j endwhile2       # break

elseifw2:
    mov.s $f6,$f4     # xlow=low
    mov.s $f7,$f5     # xhigh=high

notifw2:
    j iniwhile2

endwhile2:
    addi $t0,$0,1     # i=1

iniforw1:
    slt $t1,$t0,$t9   # i<times ?
    beq $t1,$0,endforw1

    addi $t0,$0,10    # t0=10
    mtc1 $t0,$f16     # f16=10
    cvt.s.w $f16,$f16 # f16=10.0
    div.s $f8,$f8,$f16 # sqrt=sqrt/10

    addi $t0,$t0,1    # i++
    j iniforw1

endforw1:
    mov.s $f0,$f8     # prepare return sqrt

    j endsqrt1        # skip retmeno1 code below

retmeno1:
    addi $t0,$0,-1    # t0=-1
    mtc1 $t0,$f0      # f0=-1
    cvt.s.w $f0,$f0   # f0=-1.0
    j endsqrt1

endsqrt1:

    jr $ra
```

Console

```
2.64575148
5.19615269
```

ESERCIZIO 2

```
A = 4
B = 8
C = 128
RP = LRU
Thit = 4
Tpen = 40
File: c1180214.sh_001000.din
Read 22 references.
=== T    X   XM  XT  XS  XB  H [SET]:USAGE [SET]:MODIF  [SET]:TAG
=== R  127   15   3   3   7  0 [3]:3,0,0,0 [3]:0,0,0,0  [3]:3,-,-,-
=== W  113   14   3   2   1  0 [2]:3,0,0,0 [2]:0,0,0,0  [2]:3,-,-,-
=== R  163   20   5   0   3  0 [0]:3,0,0,0 [0]:0,0,0,0  [0]:5,-,-,-
=== W  111   13   3   1   7  0 [1]:3,0,0,0 [1]:0,0,0,0  [1]:3,-,-,-
=== R  140   17   4   1   4  0 [1]:2,3,0,0 [1]:0,0,0,0  [1]:3,4,-,-
=== W  161   20   5   0   1  1 [0]:3,0,0,0 [0]:1,0,0,0  [0]:5,-,-,-
=== R  115   14   3   2   3  1 [2]:3,0,0,0 [2]:0,0,0,0  [2]:3,-,-,-
=== W  224   28   7   0   0  0 [0]:2,3,0,0 [0]:1,0,0,0  [0]:5,7,-,-
=== R  222   27   6   3   6  0 [3]:2,3,0,0 [3]:0,0,0,0  [3]:3,6,-,-
=== W  241   30   7   2   1  0 [2]:2,3,0,0 [2]:0,0,0,0  [2]:3,7,-,-
=== R  216   27   6   3   0  1 [3]:2,3,0,0 [3]:0,0,0,0  [3]:3,6,-,-
=== W  313   39   9   3   1  0 [3]:1,2,3,0 [3]:0,0,0,0  [3]:3,6,9,-
=== R  416   52  13   0   0  0 [0]:1,2,3,0 [0]:1,0,0,0  [0]:5,7,13,-
=== W  523   65  16   1   3  0 [1]:1,2,3,0 [1]:0,0,0,0  [1]:3,4,16,-
=== R  691   86  21   2   3  0 [2]:1,2,3,0 [2]:0,0,0,0  [2]:3,7,21,-
=== W  716   89  22   1   4  0 [1]:0,1,2,3 [1]:0,0,0,0  [1]:3,4,16,22
=== R  831  103  25   3   7  0 [3]:0,1,2,3 [3]:0,0,0,0  [3]:3,6,9,25
=== W  910  113  28   1   6  0 [1]:3,0,1,2 [1]:0,0,0,0  [1]:28,4,16,22 (out: XM=13 XT=3  XS=1 )
=== R 1011  126  31   2   3  0 [2]:0,1,2,3 [2]:0,0,0,0  [2]:3,7,21,31
=== W 1118  139  34   3   6  0 [3]:3,0,1,2 [3]:0,0,0,0  [3]:34,6,9,25 (out: XM=15 XT=3  XS=3 )
=== R 1231  153  38   1   7  0 [1]:2,3,0,1 [1]:0,0,0,0  [1]:28,38,16,22 (out: XM=17 XT=4  XS=1 )
=== W  121   15   3   3   1  0 [3]:2,3,0,1 [3]:0,0,0,0  [3]:34,3,9,25 (out: XM=27 XT=6  XS=3 )
--------------------------------------
P1 Nmiss=19   Nhit=3  Nref=22  mrate=0.863636   AMAT=38.5455
```

COMPITO di ARCHITETTURA DEI CALCOLATORI del 14-02-2018 (rev1.1) **MATRICOLA**_____

SOLUZIONE **COGNOME**_____

NOME_____

ESERCIZIO 3

```
lab: LW  $2, 0($1)      FORMATO I (op,rs,rd,im):        23  1  2  0000
     SW  $4, 0($3)      FORMATO I (op,rs,rd,im):        2B  3  4  0000
     SLL $5, $2, 10     FORMATO R (op,rs,rt,rd,sh,fu): 00  2  0  5 0A 00
     SRA $6, $4, 12     FORMATO R (op,rs,rt,rd,sh,fu): 00  4  0  6 0C 03
     MULTU   $5, $6     FORMATO R (op,rs,rt,rd,sh,fu): 00  5  6  0 00 19
     MFLO    $7         FORMATO R (op,rs,rt,rd,sh,fu): 00  0  0  7 00 12
     JAL fun            FORMATO J (op,im)              03  10000A          # indirizzo(fun)=400028 → (400028/4)=10000A
     SLT $8, $7, $2     FORMATO R (op,rs,rt,rd,sh,fu): 00  7  2  8 00 2A
     BNE $8, $0, lab    FORMATO I (op,rs,rd,im):        05  8  0  FFF7      # branch di -9 istruzioni
     JR  $31            FORMATO R (op,rs,rt,rd,sh,fu): 00 1F  0  0 00 08
fun: DIV $10, $12       FORMATO R (op,rs,rt,rd,sh,fu): 00  A  C  0 00 1A
     MFHI  $2           FORMATO R (op,rs,rt,rd,sh,fu): 00  0  0  2 00 10
     JR    $31          FORMATO R (op,rs,rt,rd,sh,fu): 00 1F  0  0 00 08
```

ESERCIZIO 4

```
`timescale 1ns/1ps
module TopLevel;
  reg clock; reg reset_;
  wire[3:0] Q;
  always #10 clock<=(!clock);
  initial begin
          $display ("time, \t clock, \t reset_, \t QQQQ");
          $monitor ("%g, \t %b, \t %b, \t %b", $time, clock, reset_, Q);
          reset_ = 1'b1;
          clock = 0;
          #5 reset_ = 1'b0;
          #20 reset_ = 1'b1;
          #600 $finish;
  end
  reg T; initial begin T=1; #650 T=0; end
  Serial_Carry_Counter src(Q,T,clock,reset_);
  //debug:
  wire q0=src.q0, q1=src.q1, q2=src.q2, q3=src.q3;
  wire[3:0] q= {src.q3,src.q2,src.q1,src.q0};
endmodule

// Flip-Flop T sensibile al fronte in discesa con riporto
module FFTnr(q,tin,tout,clock,reset_);
  input clock, reset_;
  input tin;
  output q,tout;
  reg STAR;
  parameter S0=0,S1=1;
  assign q=(STAR==S0)?0:1;
  assign tout=tin & q;
  always @(reset_==0) #1 STAR <= S0;
  always @(negedge clock) if (reset_==1) #3
    casex(STAR)
      S0: STAR <= (tin==0)?S0:S1;
      S1: STAR <= (tin==1)?S0:S1;
    endcase
endmodule

module Serial_Carry_Counter(Q,T,clock,reset_);
  input clock, reset_,T;
  wire q0,q1,q2,q3,to0,to1,to2,to3;
  reg[3:0] Q1;
  output[3:0] Q; assign Q=Q1;
  FFTnr sc0(q0,  T,to0,clock,reset_);
  FFTnr sc1(q1,to0,to1,clock,reset_);
  FFTnr sc2(q2,to1,to2,clock,reset_);
  FFTnr sc3(q3,to2,to3,clock,reset_);
  //notare che per non avere stati spuri in uscita conviene bufferizzare
in un registro
  always @(negedge clock) begin Q1[0]<=q0; Q1[1]<=q1; Q1[2]<=q2;
Q1[3]<=q3; end
endmodule
```