

MOV R1, #43

MOV R2, #50


Scambio DI REG → MOV R2, R1  
ECC

ADD DEST, OP1, OP2

↓                      ↓                      ↓

REG                      REG                      NVN / REG

ADD R0, R1<sup>+</sup>, R2

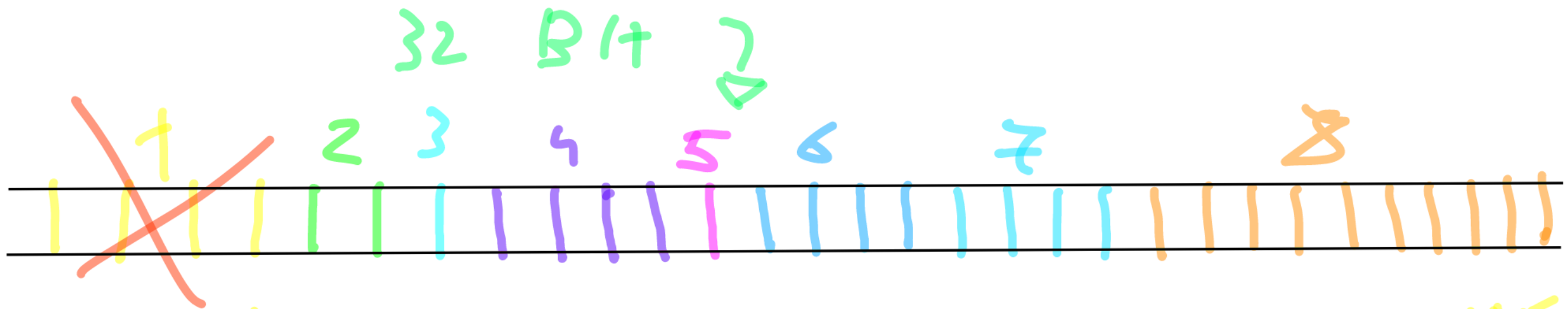


Solnko  $R1 + R2$  is stored in R0

SUB DEST, OP1, OP2

OpM 15th  $\rightarrow$  30 Bit



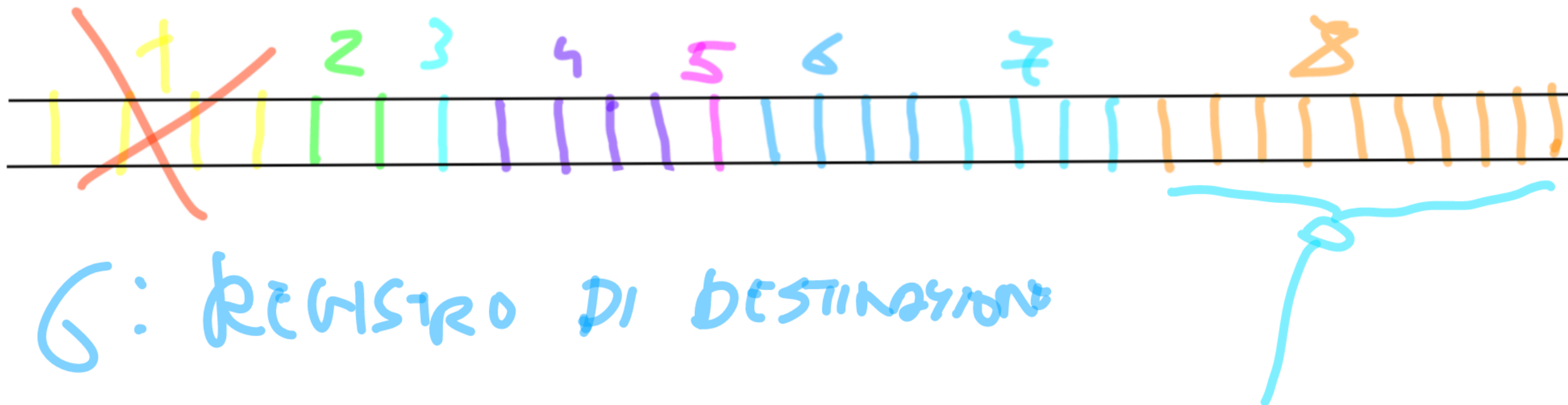


1: CONDITION FIELD → ESCUZIONE CONDIZIONALE ISTRUZIONI  
 ↳ NON MEDIANO

3: 1 → IMMEDIATO (NUMERO) → CODIFIED SECONDO OPERANDO  
 0 → REGISTRO

4: ISTRUZIONE E SELEZIONE →  $2^N$  →  $2^h = 16$  → 0000  
 0001 ECC  
 0010

5: SET CONDITION CODES → + → L'OP PUBBL  
 MODIFICARE I CONDITION CODES



6: REGISTRO DI DESTINAZIONE

7: PRIMO OPERANDO → CAMBIO SE È  
VIL REGISTRO/NUMERO



ADD R0, R0, R1

1 2 3 4 5 6 7 8  
 1110 | 00 | 0 | 0100 | 0 | 0000 | 0000 | 00000001

ADD → DUE REGISTRI

Quello  
 che andiamo  
 a sommare

2: ISTRUZIONE ARITMETICA

3: 0 PERCHÉ È LA OPERATION TRA REGISTRI

4: 0100 → TIPO DI OPERANDI → ADD

6: SECONDA OPERANDA: REGISTRO 0 → DESTINAZIONE

7: PRIMO OPERANDA → REGISTRO 1

1 2 3 4 5 6 7 8  
1 1 1 0 | 0 0 | 0 | 0 1 0 0 | 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 0 0 0 0 1 1

ADD R0, R0, R1  
C0 SORGENTI  
C1 REGISTRO  
RET DI DESTINAZIONE  
SECONDO  
OPERANDO

1 2 3 4 5 6 7 8  
1 1 1 0 | 0 0 | 0 | 0 1 0 0 | 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 0 0 0 0 1 1



ADD R1, R2, #2

(SALO REGISTRI(13))

1 2 3 4 5 6 7 8 9  
1110 | 0011 | 0100 | 0100 | 0010 | 0010 | 0001 | 0000 | 0000 10

} : SECONDO OPERANDO

DIVERSE

DE PRIMA

↳ COSTRUTTORE

4 : 0100 → ADDIZIONE

9 : VALORE 2 → 8 BIT → 0000 0010

6 : REGISTRO 2 (R2) → SORENTE

7 : REGISTRO 1 (R1) → DESTINAZIONE

2<sup>1</sup>

MOV R1, #8 → Metto 8 NEL REG R1

CMP R1, #0 → COMPARAZIONE → Z → 1 SI

BEQ FINE → SE CMP TORNO 1 → 0 NO

ADD R1, R1, #1

FINE:

ADD R1, R1, #1

SOLUZIONE DEI PRIMI 16 NUovi NATURALI

MOV R0, #0 → inizializzato a 0

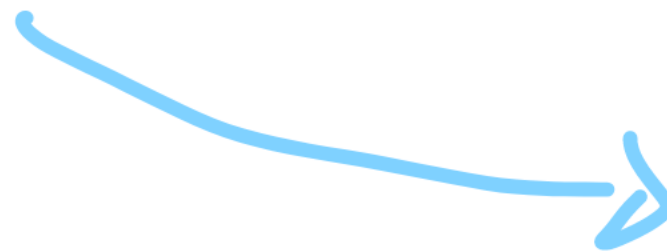
MOV R1, #1 → i

~~loop~~ ← CMP R1, #11 → BEQ FINE

SE NON È UGUALE → R0, (R0, R1)

ADD R1, R1, #1

13 LOOP  
FINE:



LOOP:

→ CMP R1, #1 • ?

BEQ FINE

ADD R0, R0, R1

ADD R1, R1, #1

→ LOOP

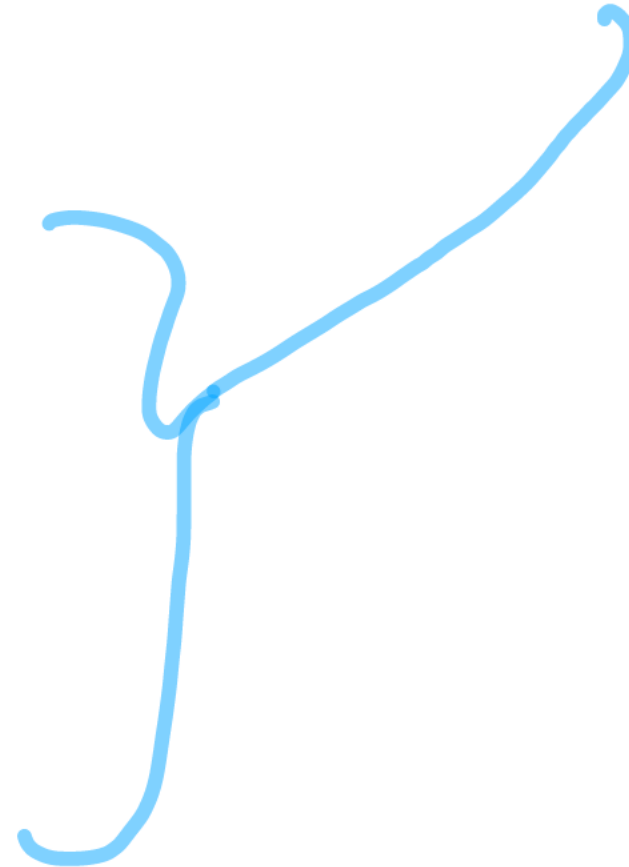
FINE:

FINISH

↓

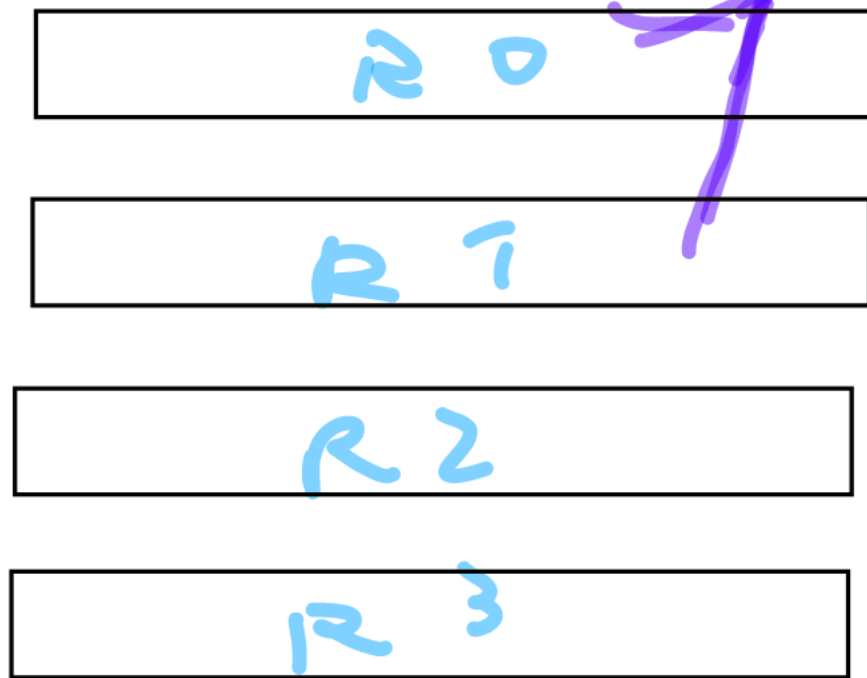
S

N



# CHIAMATE A FUNZIONE

→ VAL DI RITORNO



SALVANO A SEGUENTI  
NEI REGISTRI






LO STESSE ~~DELLA~~ FUNZIONE CHIAMANTE  
VIENE SALVATO IN MEMORIA

↳ R4 ..... R11

A 13

↳ BRANCH (BR)  
↳ VBL STACK  
POINTER

ARUDMENT: DA  
PASSWER BUCH  
FUNKTION

RTN

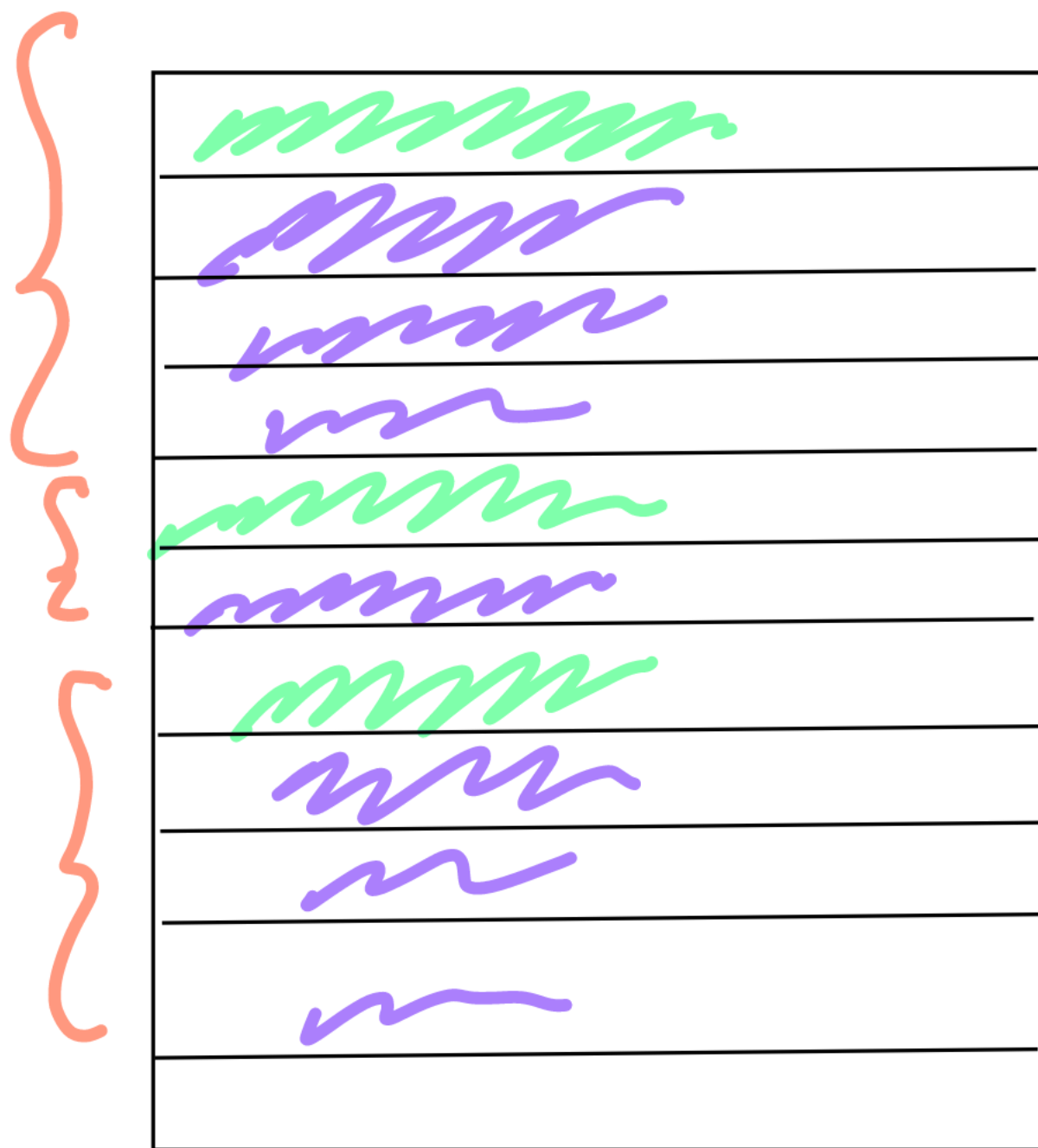
INDIRIZZ  
DIRIT

Stack  
FRAME

— Stack Frame

— RETURN (ADDRESS)

— VAR  
LOCAL



→ R+3