Lezione 10: L'organizzazione a pila

Elementi di Architettura e Sistemi Operativi
Docente: Tiziano Villa

Corso di Laurea in Bioinformatica

Gennaio 2014

Argomenti della lezione

- La struttura dati detta pila (stack).
- Le interruzioni (rivisitazione).
- Aritmetica con la pila.

Fonte:
Patt & Patel: "*Introduction to Computing Systems: From Bits and Gates to C and Beyond*". Ch. 10.

Il concetto di pila (stack)

- Le strutture dati astratte si definiscono mediante le regole per inserire e rimuovere dati.
- Le regole per definire una pila sono:
- Ultimo inserito, primo rimosso (Last In, First Out, LIFO).
  - Si dice che si **infila (push)** un elemento *sulla* pila quando si aggiunge.
  - Si dice che si **sfila (pop)** un elemento *dalla* pila quando si rimuove.

Analogia con una pila di libri:

- Si pone un nuovo libro in cima alla pila.
- Si puo' rimuovere un libro solo dalla cima della pila.
- L'elemento (libro) aggiunto per ultimo alla pila sara' il primo ad essere rimosso dalla pila.

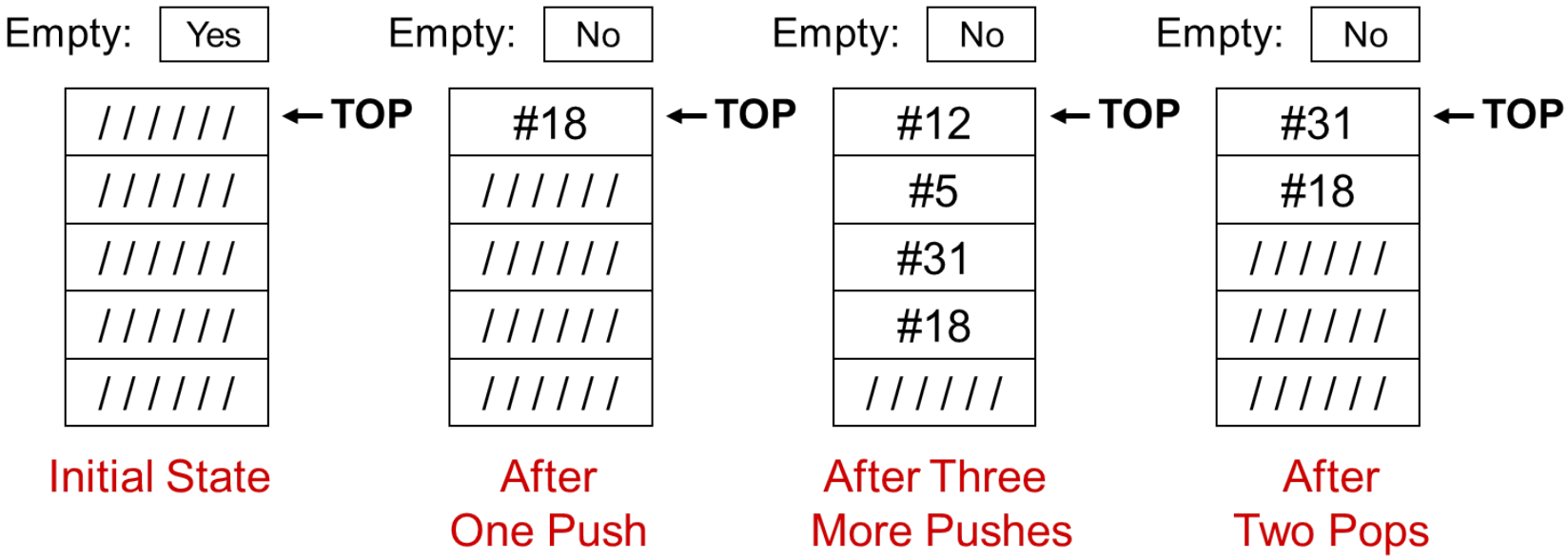
Realizzazione della pila

Richiede 4 operazioni:

- **Push**: pone un valore sulla pila.
- **Pop**: rimuove un valore dalla pila.
- **Is_Empty**: verifica se la pila e' vuota.
- **Is_Full**: verifica se la pila e' piena.

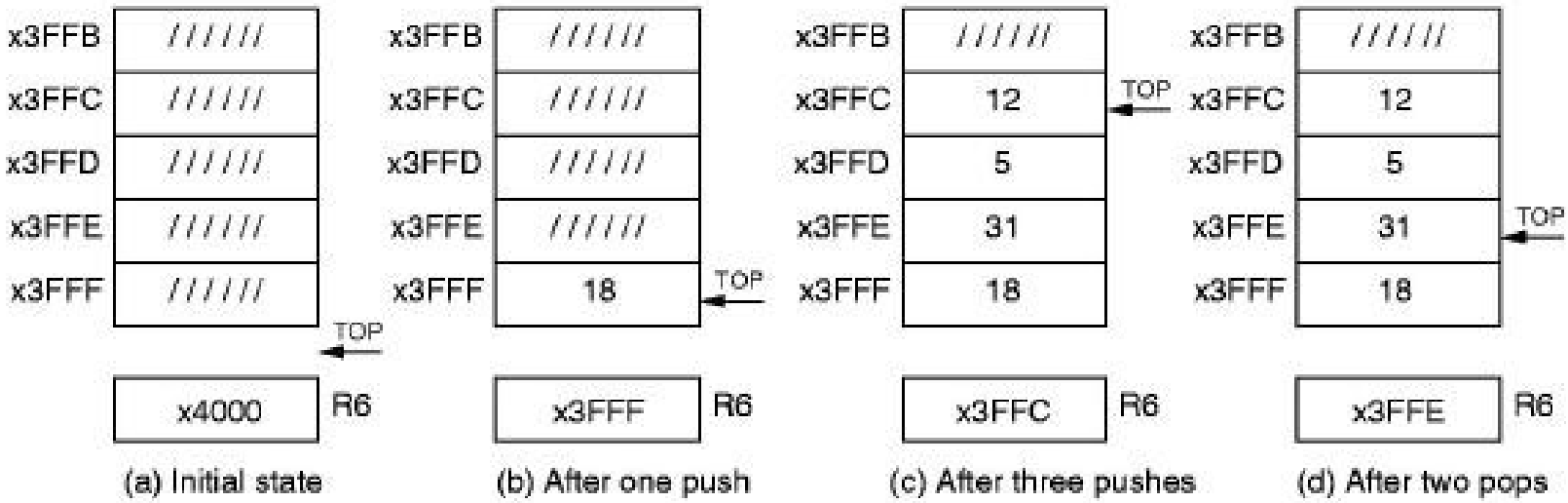
Realizzazione della pila con registri

- I dati precedenti si muovono all'ingiu' o all'insu' da registro a registro per far posto a ogni nuovo inserimento o rimozione.
- Il puntatore TOS punta sempre alla medesima posizione.



Realizzazione della pila in memoria

- Il puntatore alla cima della pila (TOS) si sposta quando si aggiungono o si rimuovono dati.
- Per convenzione il puntatore alla cima della pila (TOS) e' contenuto nel registro R6.



Semantica di Push e Pop

Push:

- Si decrementa il puntatore TOS (la nostra pila si muove verso il basso).
- Si trascrive il dato in R0 nella posizione a cui punta il nuovo TOS.

Pop:

- Si trascrive in R0 il dato a cui punta il TOS corrente.
- S'incrementa il puntatore TOS.

Codice di LC-3 per Push e Pop

Push

```
ADD   R6, R6, #-1 ; decrement stack ptr
STR   R0, R6, #0  ; store data (R0)
```

Pop

```
LDR   R0, R6, #0  ; load data from TOS
ADD   R6, R6, #1  ; increment stack ptr
```

Codice di LC-3 per Push e Pop

Che cosa succede se la pila e' gia' piena o vuota ?

- Prima d'inserire, si deve verificare se c'e' trabocco (overflow), cioe' se cerchiamo d'infilare troppi elementi sulla pila.
- Prima di rimuovere, si deve verificare se c'e' difetto (underflow), cioe' se cerchiamo di sfilare troppi elementi dalla pila.
- In entrambi i casi si usa il registro R5 per indicare successo (R5 = 0) o insuccesso (R5 = 1).

Pop con rilevamento del difetto (underflow)

```
POP    LD  R1, EMPTY  ; EMPTY = -x4000
       ADD R2, R6, R1 ; Compare stack pointer
       BRz FAIL       ; with x4000
       LDR R0, R6, #0
       ADD R6, R6, #1
       AND R5, R5, #0 ; SUCCESS: R5 = 0
       RET
FAIL   AND R5, R5, #0 ; FAIL: R5 = 1
       ADD R5, R5, #1
       RET
EMPTY .FILL xC000
```

Push con rilevamento del trabocco (overflow)

```
PUSH  LD  R1, MAX     ; MAX = -x3FFB
      ADD R2, R6, R1  ; Compare stack pointer
      BRz FAIL        ; with x3FFB
      ADD R6, R6, #-1
      STR R0, R6, #0
      AND R5, R5, #0  ; SUCCESS: R5 = 0
      RET
FAIL  AND R5, R5, #0  ; FAIL: R5 = 1
      ADD R5, R5, #1
      RET
MAX   .FILL xC005
```

Codice LC-3 completo per gestire la pila - 0

- Per aggiungere un valore alla pila si carica tale valore in R0 e si esegue JSR PUSH.
- Per rimuovere un valore dalla pila e caricarlo in R0 si esegue JSR POP.
- Per cambiare l'inizio e la dimensione della pila si cambiano BASE e MAX (nell'esempio, la pila va da x3FFF inizio a x3FFB fine, 5 locazioni). BASE contiene -x3FFF e MAX contiene -x3FFB.
- Le procedure PUSH e POP usano i registri R1, R2, R5.
  - R1 e R2 sono salvati e ripristinati dal chiamato (il chiamante non e' neppure tenuto a sapere che sono usati nel chiamato).
  - R5 e' salvato dal chiamante (consapevole che in R5 si trova l'indicazione di successo o fallimento) prima di eseguire JSR PUSH o JSR POP.

Codice LC-3 completo per gestire la pila - 1

```
POP     ST    R2, Sv2          ;save, needed by POP
        ST    R1, Sv1          ;save, needed by POP
        LD    R1, BASE         ;BASE contains x-3FFF
        ADD   R1,  R1, # -1    ;R1 now has x-4000
        ADD   R2, R6, R1       ;Compare SP to x4000
        BRz   fail_exit        ;Branch if stack is empty
        LDR   R0, R6, # 0      ;The actual 'pop'
        ADD   R6, R6, # 1      ;Adjust stack pointer
        BRnzp success_exit
```

Codice LC-3 completo per gestire la pila - 2

```
PUSH    ST    R2, Sv2          ;needed by PUSH
        ST    R1, Sv1          ;needed by PUSH
        LD    R1, MAX          ;MAX has x-3FFB
        ADD   R2, R6, R1       ;Compare SP to x3FFB
        BRz   fail_exit        ;Branch is stack is full
        ADD   R6, R6, # -1     ;Adjust Stack Pointer
        STR   R0, R6, # 0      ;The actual ‘push’
```

Codice LC-3 completo per gestire la pila - 3

```
success_exit  LD    R1, Sv1      ;Restore register values
              LD    R2, Sv2      ;
              AND   R5, R5, # 0  ;R5 <-- success
              RET
              ;
fail_exit     LD    R1, Sv1      ;Restore register values
              LD    R2, Sv2
              AND   R5, R5, # 0
              ADD   R5, R5, # 1  ;R5 <-- fail
              RET

BASE          .FILL xC001              ;Base has x-3FFF
MAX           .FILL xC005              ;Max has x-3FFB
Sv1           .FILL x0000
Sv2           .FILL x0000
```

Usi della pila

- Salvare lo stato del programma quando si serve un'interruzione per una richiesta I/U: si salvano i registri PC e PSR (Processor Status Register) in una pila speciale chiamata pila del supervisore (supervisor stack).
- La pila come un'alternativa ai registri. Alcune architetture usano una pila per tutte le memorizzazioni temporanee (macchine a 0 indirizzi, in alternativa alle architetture a 3 indirizzi come LC-3).
  - L'istruzione ADD di LC-3 richiede 3 registri per gli operandi: ADD DR,SR1,SR2. Le tre locazioni sono identificate esplicitamente.
  - Per eseguire l'istruzione ADD con un'architettura a pila, si prelevano due valori dalla pila, si sommano e il risultato e' inserito in cima alla pila. Nessun operatore e' identificato esplicitamente.
  - La maggior parte dei calcolatori usano una pila per l'aritmetica, mentre i microprocessori usano un banco di registri.

Rivisitazione delle interruzioni

- Un segnale d'interruzione esterno richiede d'interrompere l'esecuzione corrente, eseguire la procedura d'interruzione e alla fine riprendere il programma interrotto.
  - Se il segnale di richiesta di un'interruzione (prodotto dei segnali di pronto e abilitazione dell'interruzione, cioe' Ready e Interrupt Enable) ha una priorita' piu' alta (PL) di quella del processo in esecuzione, si asserisce il segnale INT.
- Sono rimaste aperte due domande dalla precedente introduzione alla gestione dell'ingresso e uscita con le interruzioni:
  - Come salvare informazione sufficiente circa il programma in esecuzione per poterlo continuare quando si e' finito di servire l'interruzione ?
  - Come ri-inizializzare il processore per gestire la procedura di servizio dell'interruzione ?

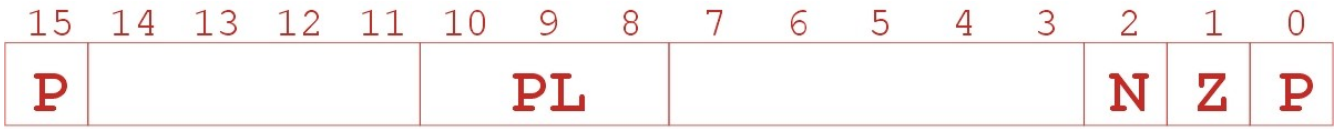
Lo stato di un processore

- Lo stato e' una fotografia instantanea di un sistema. Include il contenuto della memoria rilevante, i registri generali, e alcuni registri speciali come il contatore di programma (PC) e il registro di stato del processore (PSR).
- Nell'architettura di LC-3 e' sufficiente salvare PC e PSR. Non e' necessario salvare i registri generali perche' si assume che il loro salvataggio e ripristino sara' un compito della procedura di servizio chiamata.

Il registro PSR

Che cosa contiene il PSR ?

- I **privilegi** (PSR[15], indica se il programma esegue in modo supervisore (0) o utente (1)).
- Le **priorita'** (PS[10:8] indica il livello di priorita' da PL0 piu' basso a PL7 piu' alto).
- I **codici di condizione** del programma in esecuzione (PR[2:0], codici NZP).



Dove si salva lo stato di un processore ?

- Non si possono usare i registri, perche' il programmatore non sa quando avverra' l'interruzione e percio' non e' in grado di salvare i registri rilevanti.
- Bisogna essere in grado di gestire interruzioni annidate l'una nell'altra.
- **Soluzione**: si usa un'area di memoria, anch'essa gestita come una pila, dove s'inserisce lo stato per salvarlo e lo si preleva per ripristinarlo. E' la **pila del supervisore**.

La pila del supervisore

- La pila del supervisore e' un'area speciale di memoria usata come pila per le procedure di servizio delle interruzioni.
- Esistono una pila dell'utente e una del supervisore in aree separate della memoria.
  - Se adesso non in uso, si memorizza il puntatore alla pila del supervisore (Supervisor Stack Pointer, SSP) in Saved.SSP.
  - Se adesso non in uso, si memorizza il puntatore alla pila dell'utente (User Stack Pointer, USP) in Saved.USP.
  - Il puntatore alla pila corrente si trova sempre in R6. Se il programma corrente e' in modo privilegiato, R6 punta alla pila del supervisore, altrimenti punta alla pila dell'utente.

Come si salva lo stato del programma chiamate

- Che cosa fa il processore quando riceve un segnale d'interruzione ? (Si noti che il processore verifica se c'e' un'interruzione alla fine di ogni ciclo di esecuzione di un'istruzione.)
  - Cambia il modo di esecuzione da utente a supervisore: PSR[15] = 0.
  - Salva il puntatore alla pila dell'utente e carica il puntatore alla pila del supervisore: Saved.USP = R6; R6 = Saved.SSP.
  - Salva in cima alla pila i registri PC e PSR del programma interrotto. Ora il processore puo' gestire la richiesta del dispositivo che ha interrotto l'esecuzione corrente.

Come si carica lo stato della procedura di servizio

- Il dispositivo d'I/U trasmette il segnale d'interruzione INT, il suo livello di priorita', e un vettore a 8 cifre binarie (INTV).
- Se l'interruzione e' accettata, si espande INTV a 16 cifre binarie.
  - La tavola dei vettori d'interruzione (Interrupt Vector Table) risiede in memoria nelle locazioni da x0100 a x01FF e memorizza gl'indirizzi iniziali delle procedure di servizio delle interruzioni (similmente alla tavola dei vettori delle eccezioni Trap Vector Table).
  - L'estensione di INTV e' un indice (x0100 + Zext(INTV)) nella tavola dei vettori d'interruzione, il cui elemento cosi' indicizzato contiene l'indirizzo della procedura di servizio richiesta. Si carica tale indirizzo della procedura richiesta nel PC.
  - S'inizializza il PSR come segue: PSR[15] = 0 (modo supervisore); PSR[10:8] = priorita' del chiamato; PSR[2:0] = 000 (codici azzerati).

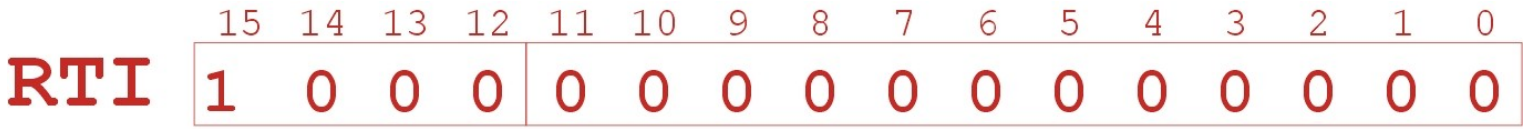
Sommario delle operazioni per gestire un'interruzione

1. Se PSR[15] = 1 (modo utente), Saved.USP = R6, R6 = Saved.SSP.
2. Inserisci PSR e PC nella pila del supervisore.
3. PSR[15] = 0 (modo supervisore).
4. PSR[10:8] = priorita' dell'interruzione in corso di servizio.
5. PSR[2:0] = 000.
6. MAR = x0100 + Zext(INTV), dove INTV = vettore d'interruzione a 8 cifre fornito dal dispositivo che interrompe (es., per la tastiera e' x80).
7. Memorizza nel registro MDR il contenuto di M[MAR].
8. Carica in PC il contenuto di MDR (indirizzo della prima istruzione della procedura di servizio dell'interruzione).

Semantica dell'istruzione RTI (ritorno dall'interruzione)

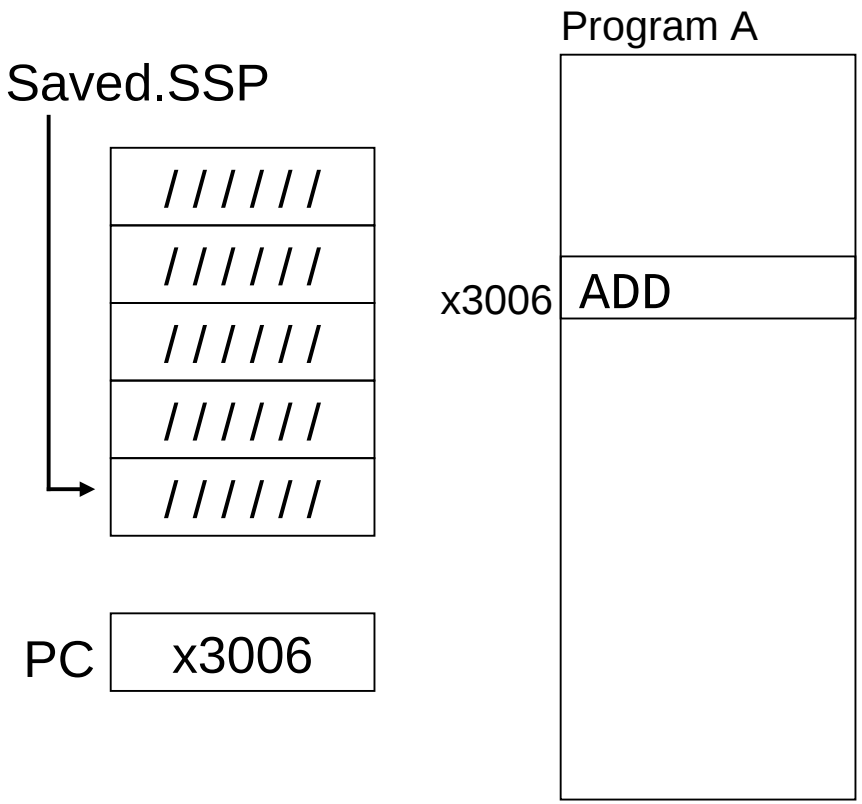
RTI e' un'istruzione che ripristina lo stato e si puo' eseguire solo in modalita' supervisore (altrimenti provoca un'eccezione).

1. PC = M[R6]; R6 = R6+1 (ripristina il PC ed eliminalo dalla pila del supervisore).
2. PSR = M[R6]; R6 = R6+1 (ripristina il PSR ed eliminalo dalla pila del supervisore).
3. Se PSR[15] = 1, Saved.SSP = R6; R6 = Saved.USP (se si ritorna in modalita' utente, salva in Saved.SSP il puntatore alla pila del supervisore e ripristina in R6 il puntatore alla pila dell'utente).



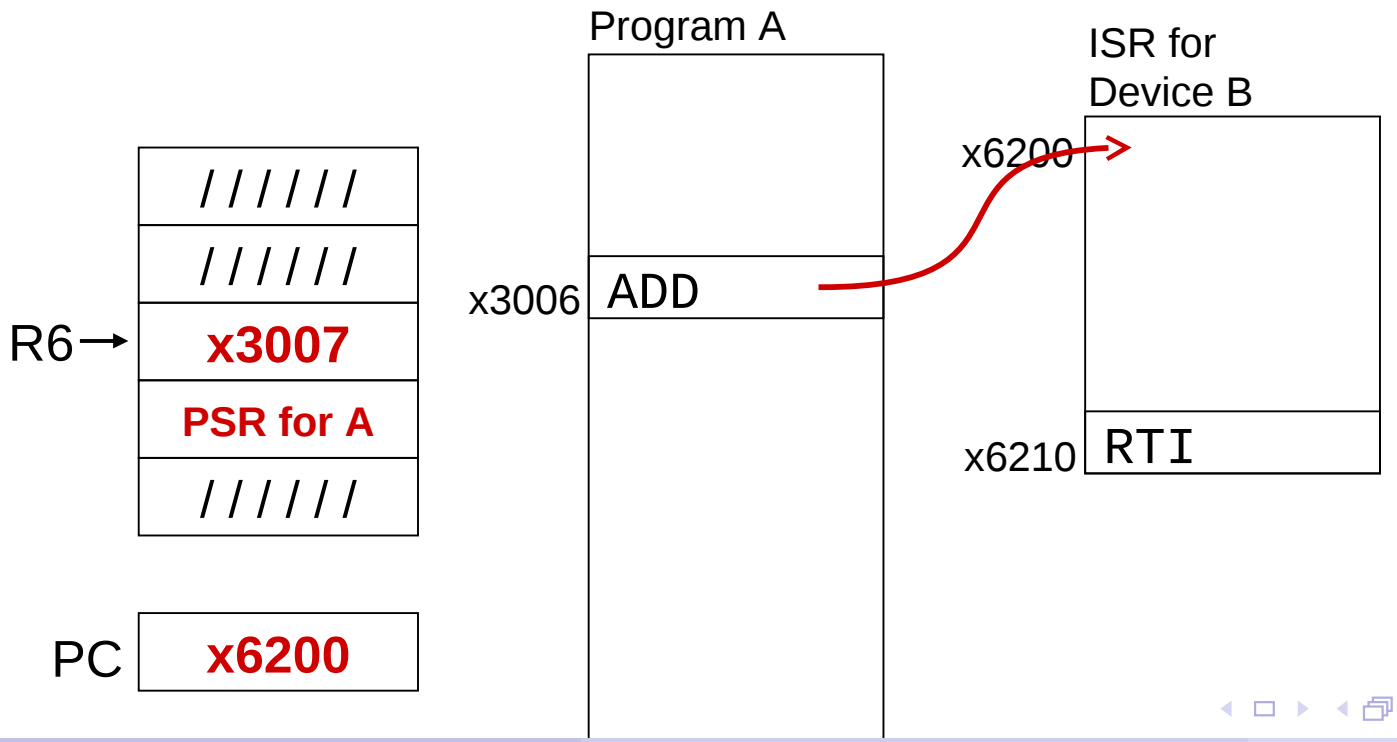
Esempio di esecuzione di un'interruzione - 1

Mentre si esegue ADD all'indirizzo x3006, arriva un'interruzione dal dispositivo B.



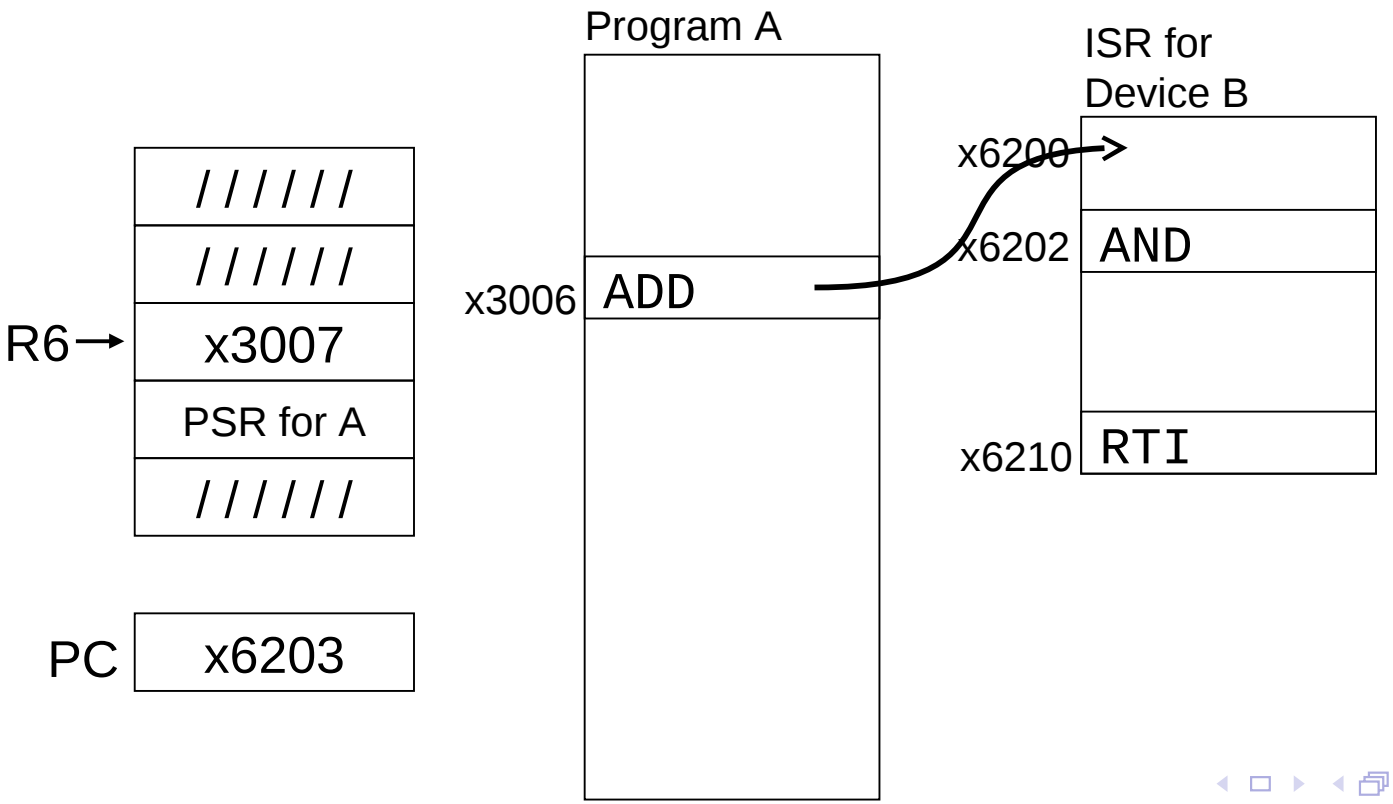
Esempio di esecuzione di un'interruzione - 2

Saved.USP = R6; R6 = Saved.SSP. Si salvano PSR e PC in cima alla pila; si esegue la procedura di servizio del dispositivo B (all'indirizzo x6200).



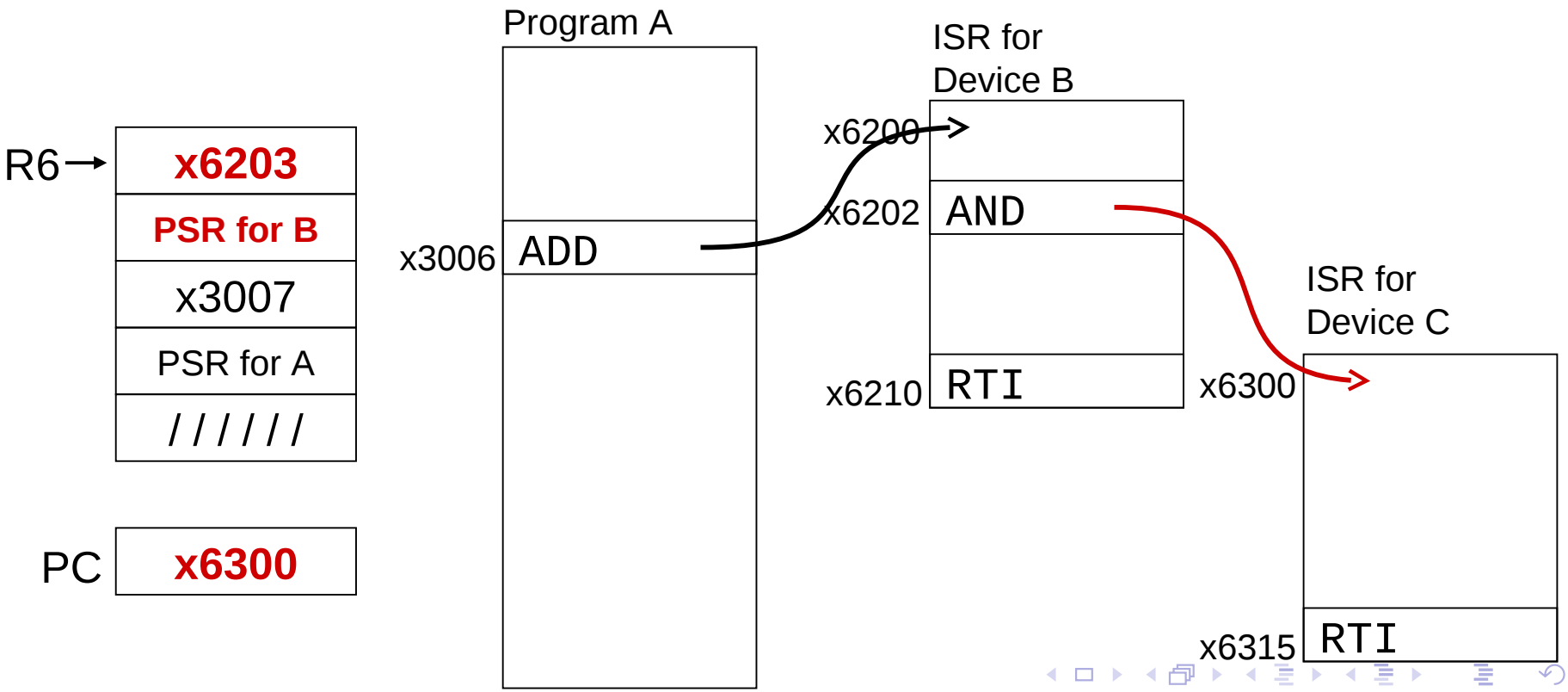
Esempio di esecuzione di un'interruzione - 3

Mentre si esegue AND all'indirizzo x6202, arriva un'interruzione dal dispositivo C.



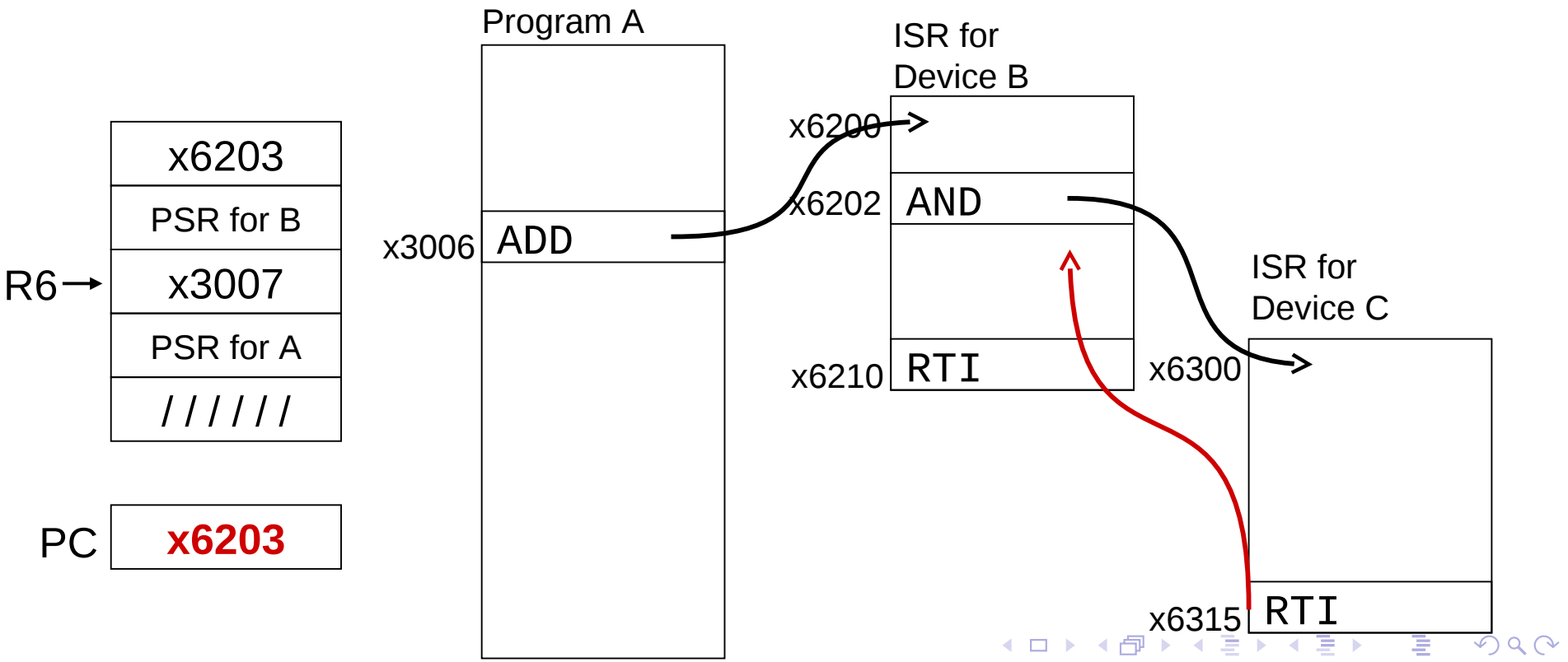
Esempio di esecuzione di un'interruzione - 4

Si salvano PSR e PC in cima alla pila, poi si esegue la procedura di servizio del dispositivo C (all'indirizzo x6300).



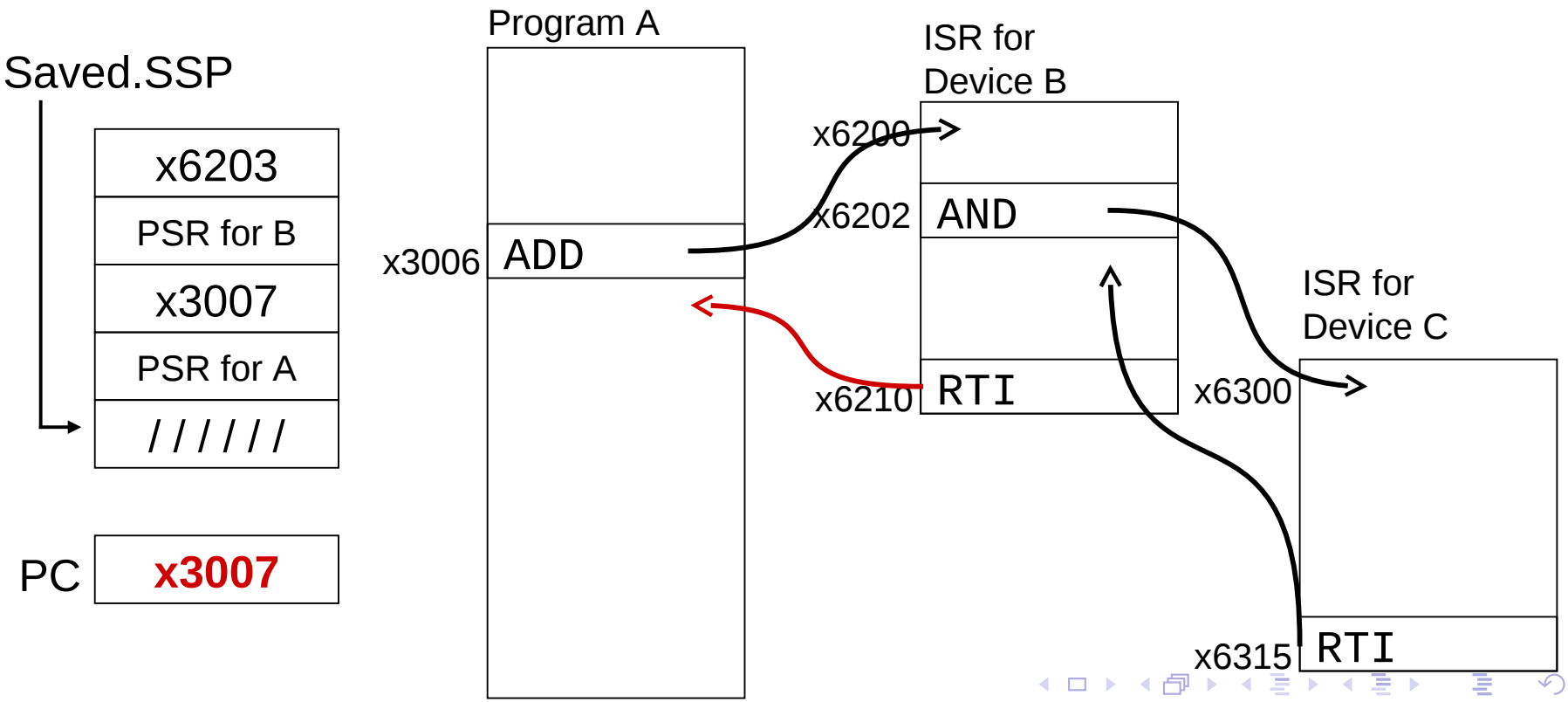
Esempio di esecuzione di un'interruzione - 5

Si esegue RTI all'indirizzo x6315; si ripristinano PC e PSR di B rimuovendoli dalla pila.



Esempio di esecuzione di un'interruzione - 6

Si esegue RTI all'indirizzo x6210; si ripristinano PC e PSR di A rimuovendoli dalla pila. Si ripristina R6 e si riprende l'esecuzione di A.



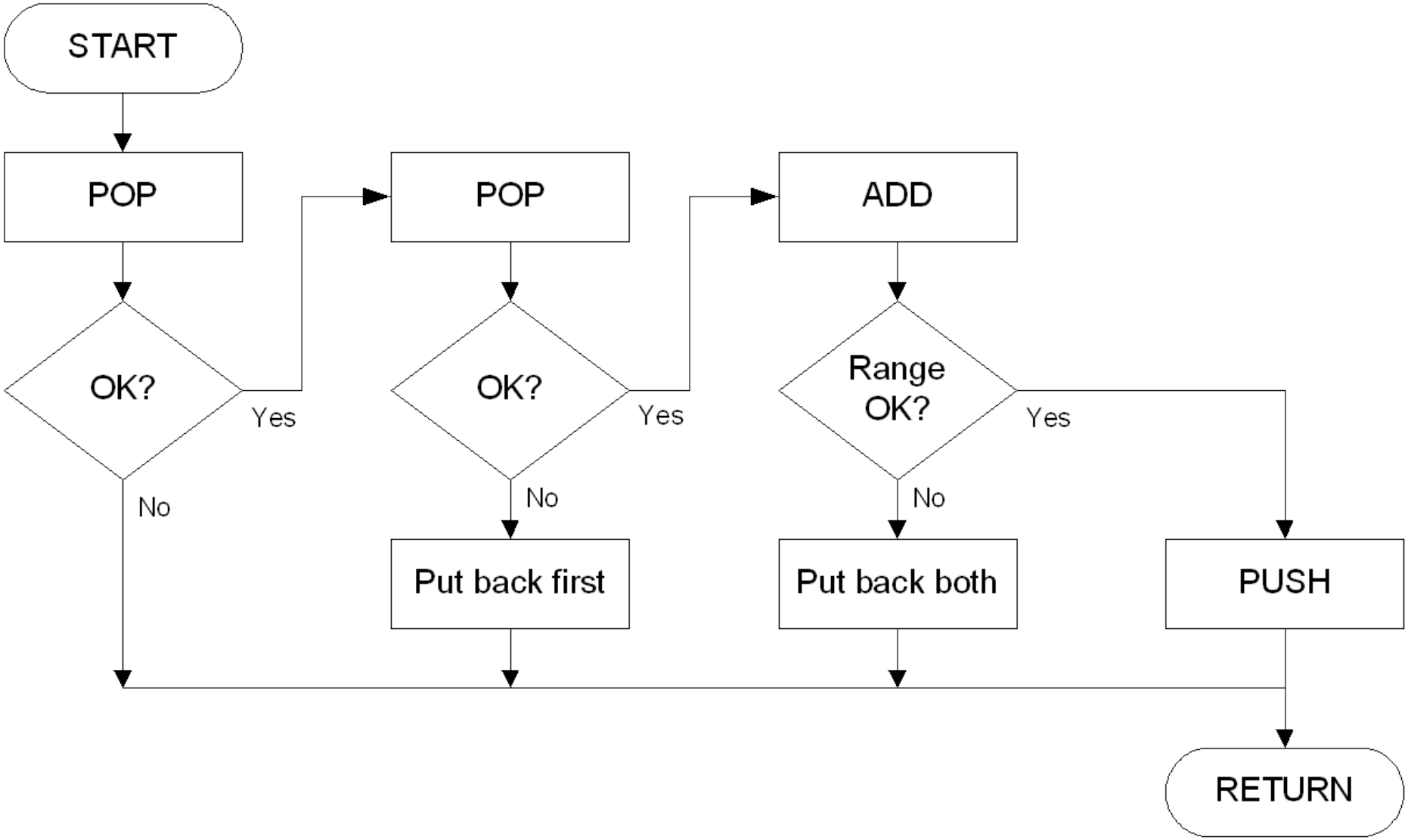
Aritmetica con la pila

Valutazione di (A+B)·(C+D) con una pila:

- push A
- push B
- **ADD**
- push C
- push D
- **ADD**
- **MULTIPLY**
- pop result

Somma eseguita con la pila

Preleva i due operandi, somma, inserisci il risultato.



Codice macchina per la somma eseguita con la pila

POP, POP, ADD, PUSH.

```
OpAdd     JSR POP          ; Get first operand.
          ADD R5,R5,#0     ; Check for POP success.
          BRp Exit         ; If error, bail.
          ADD R1,R0,#0     ; Make room for second.
          JSR POP          ; Get second operand.
          ADD R5,R5,#0     ; Check for POP success.
          BRp Restore1     ; If err, restore & bail.
          ADD R0,R0,R1     ; Compute sum.
          JSR RangeCheck   ; Check size.
          BRp Restore2     ; If err, restore & bail.
          JSR PUSH         ; Push sum onto stack.
          RET
Restore2  ADD R6,R6,#-1    ; Decr stack ptr (undo POP)
Restore1  ADD R6,R6,#-1    ; Decr stack ptr
Exit      RET
```