LABORATORIO DI ARCHITETTURA DEI CALCOLATORI

lezione n° 16

Prof. Rosario Cerbone

rosario.cerbone@libero.it

http://digilander.libero.it/rosario.cerbone

a.a. 2005-2006

SOTTOPROGRAMMI E MECCANISMI PER IL COLLEGAMENTO

Problematiche da affrontare:

Collegamento (o ***Linkage***): individuazione delle istruzioni da eseguire per realizzare le operazioni di ***chiamata*** e di ***ritorno*** delle ***procedure***

Passaggio dei Parametri**:** individuazione delle istruzioni da eseguire per realizzare il ***passaggio delle informazioni*** dal ***modulo chiamante*** alla ***procedure*** e viceversa

SOTTOPROGRAMMI E MECCANISMI PER IL COLLEGAMENTO

- Un sottoprogramma è una porzione di codice che viene eseguita mediante una chiamata, in modo da rendere il codice più semplice da scrivere e da capire.
- La chiamata viene effettuata utilizzando l'indirizzo della prima istruzione del codice del sottoprogramma, oppure, equivalentemente, il nome del sottoprogramma, che rappresenta lo stesso indirizzo.

Chiamata a Sottoprogramma

- La chiamata al sottoprogramma viene effettuata con l'istruzione:
  - JSR Indirizzo_Sottoprogramma
- mentre il ritorno al chiamante viene effettuato con l'istruzione
  - RTS

Chiamata a Sottoprogramma

- La JSR è simile ad un salto non condizionato, nel senso che salta all'indirizzo che viene passato come operando, ma differisce perché stabilisce il punto di ritorno dalla funzione.
- Infatti la JSR salva in cima allo stack il valore del Program Counter PC (4 byte), ovvero salva in cima allo stack l'istruzione che dovrà essere eseguita subito dopo la fine del sottoprogramma chiamato.
- Con questa operazione lo Stack Pointer SP viene decrementato di 4 per contenere l'indirizzo salvato.

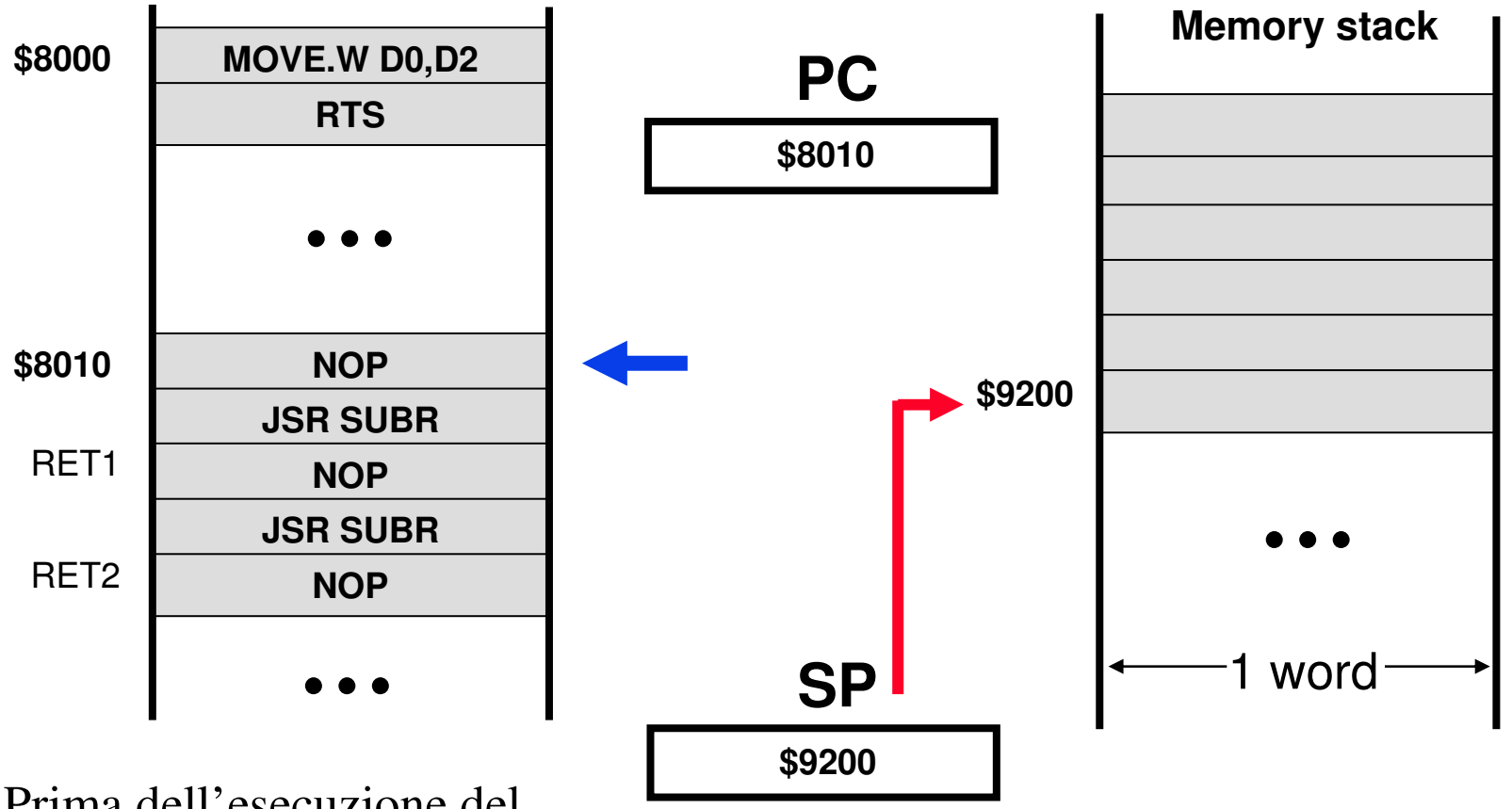
Chiamata a Sottoprogramma

- Sarà poi l'istruzione RTS a prendere dallo stack il valore salvato rimettendolo nel PC e incrementando di 4 lo SP (A7).
- Quindi dopo l'istruzione RTS il program counter avrà lo stesso valore che aveva al momento della chiamata a sottoprogramma, ed eseguirà l'istruzione successiva alla chiamata a sottoprogramma.

Esempio

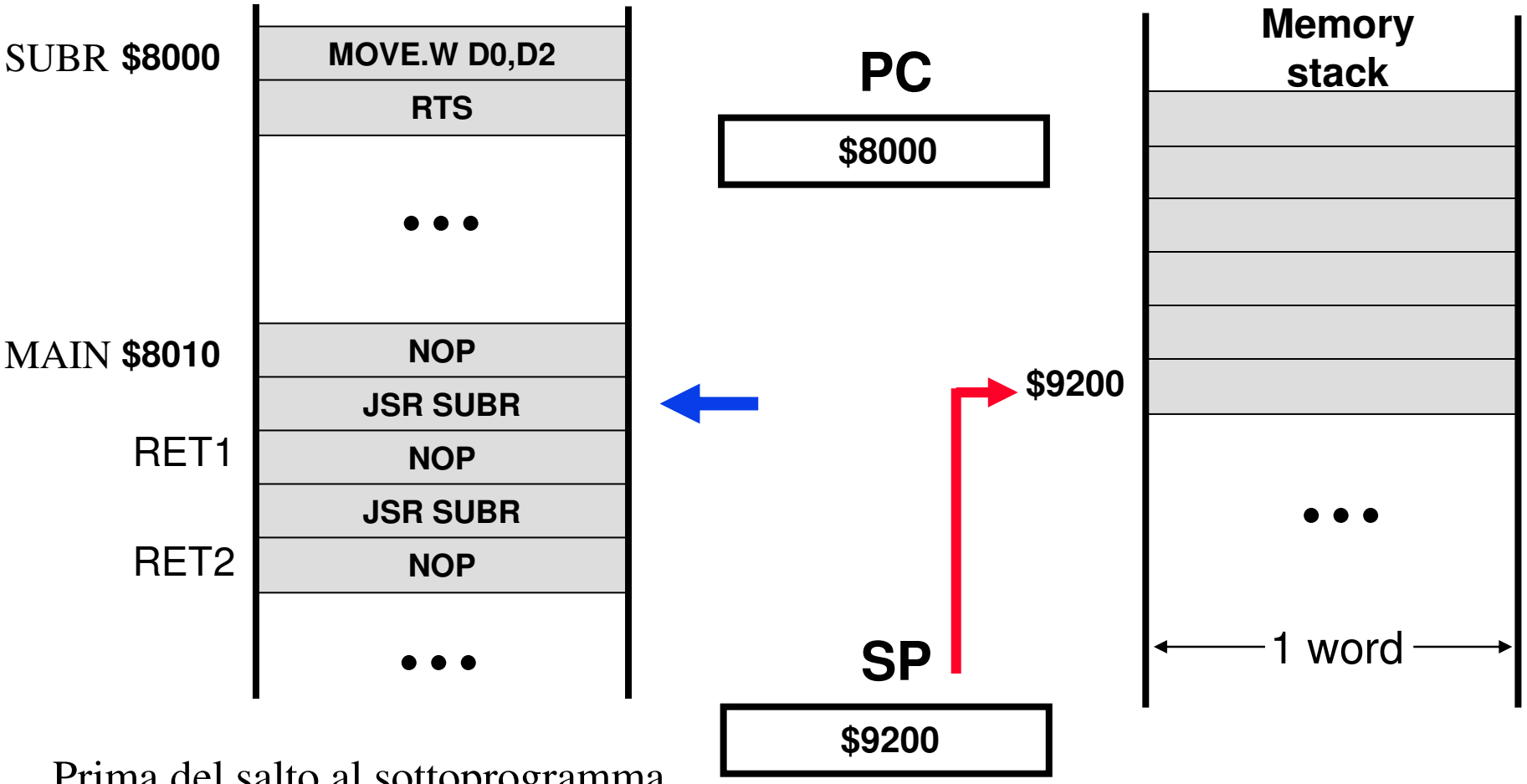
```
* Subroutine
                ORG     $8000

SUBR                    MOVE.W  D0,D2
                RTS

* Main Program
                ORG     $8010

START           NOP
                JSR     SUBR
                NOP
                JSR     SUBR
                NOP
                END     START
```

Esempio



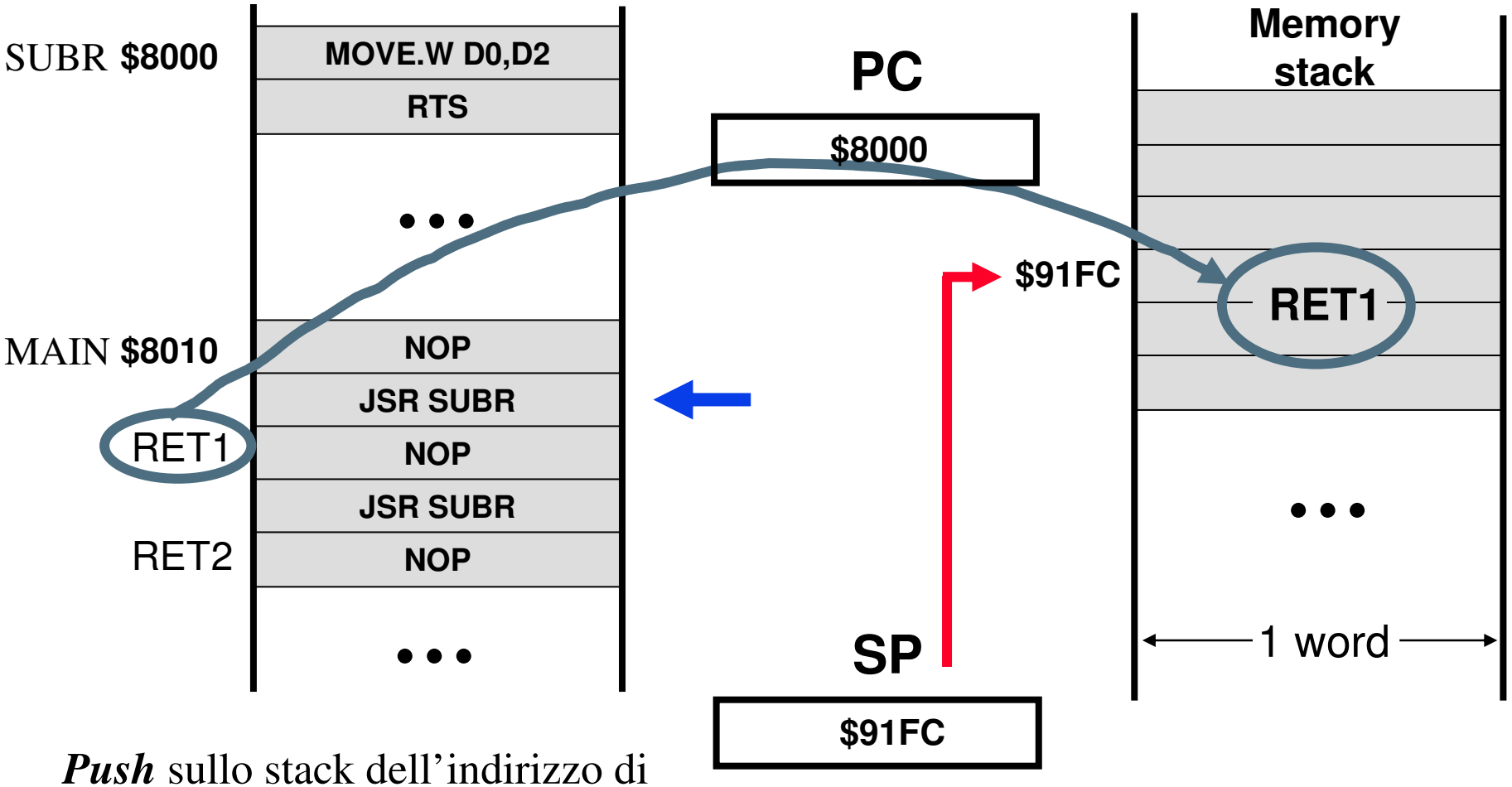
Prima dell'esecuzione del programma chiamante

Esempio - Mappa della memoria e stack



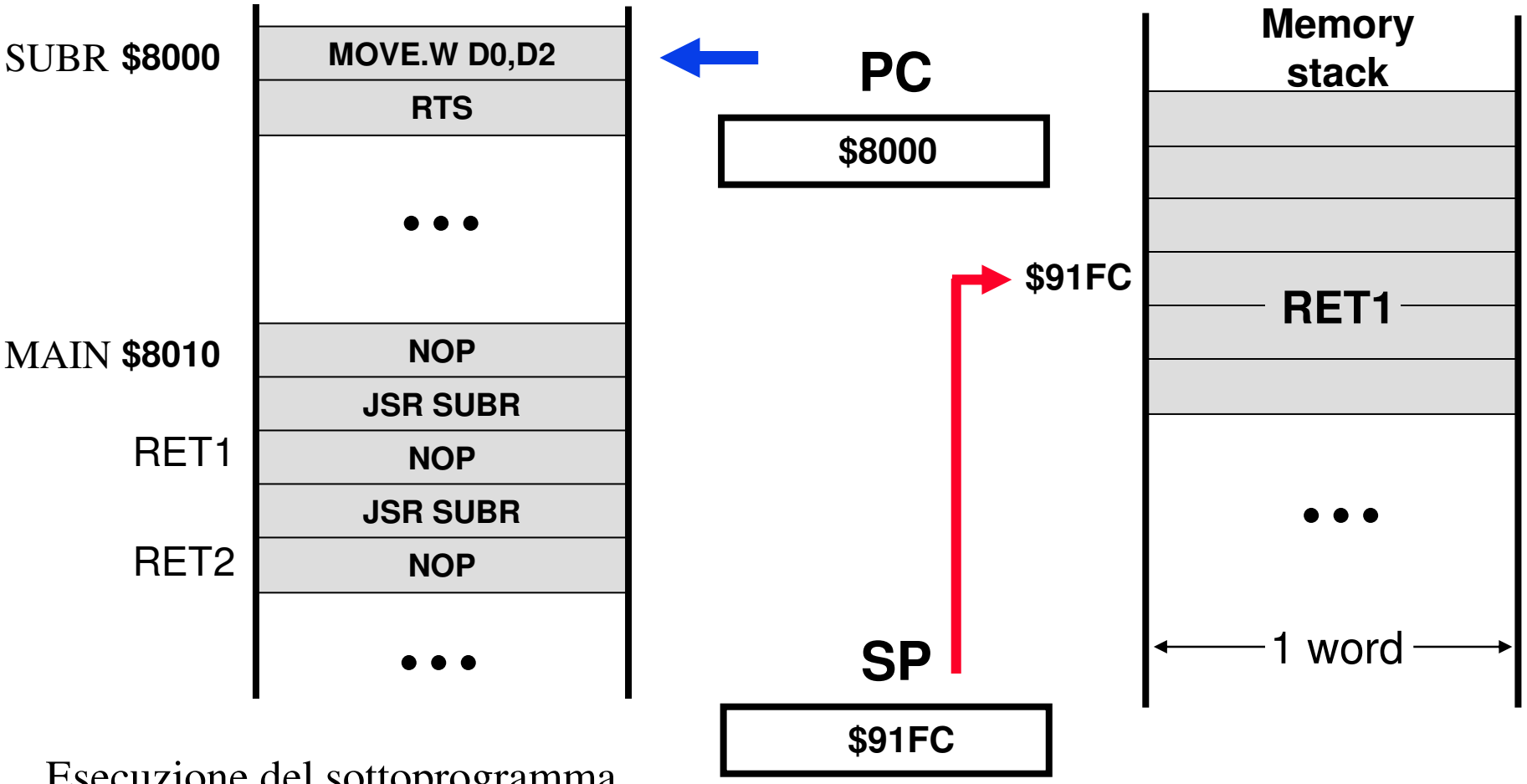
Prima del salto al sottoprogramma

Esempio - Mappa della memoria e stack



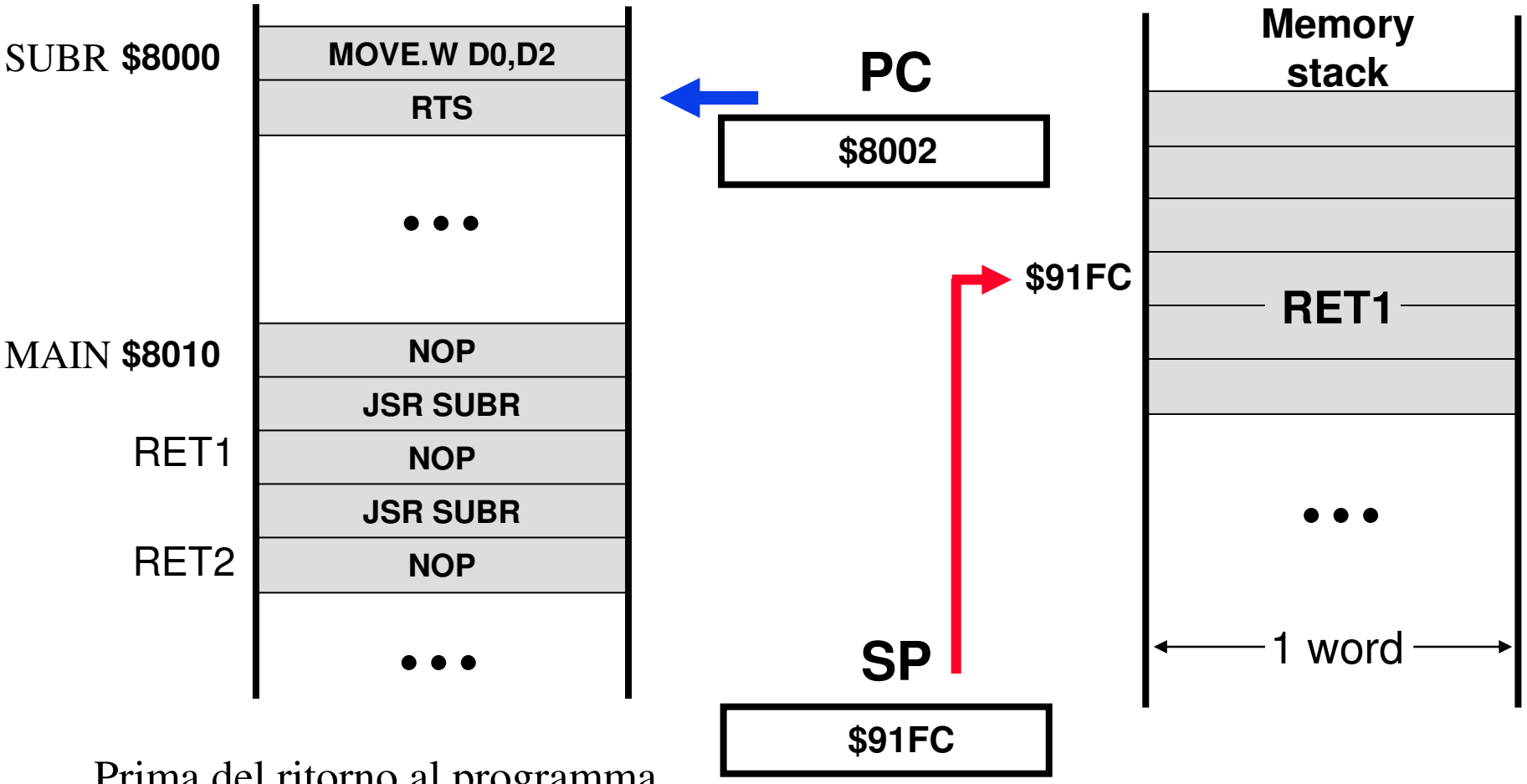
Push sullo stack dell'indirizzo di ritorno

Esempio - Mappa della memoria e stack



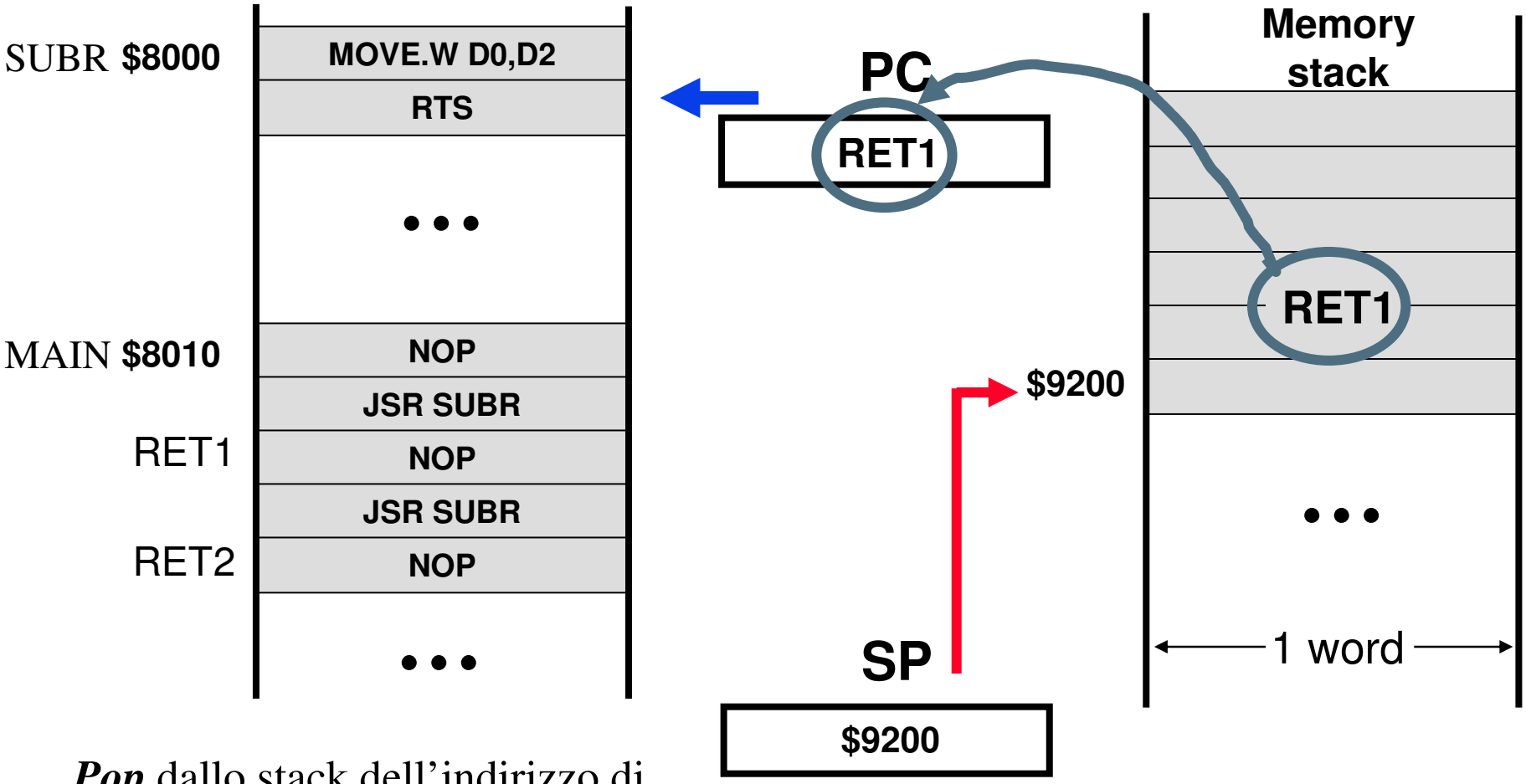
Esecuzione del sottoprogramma

Esempio - Mappa della memoria e stack



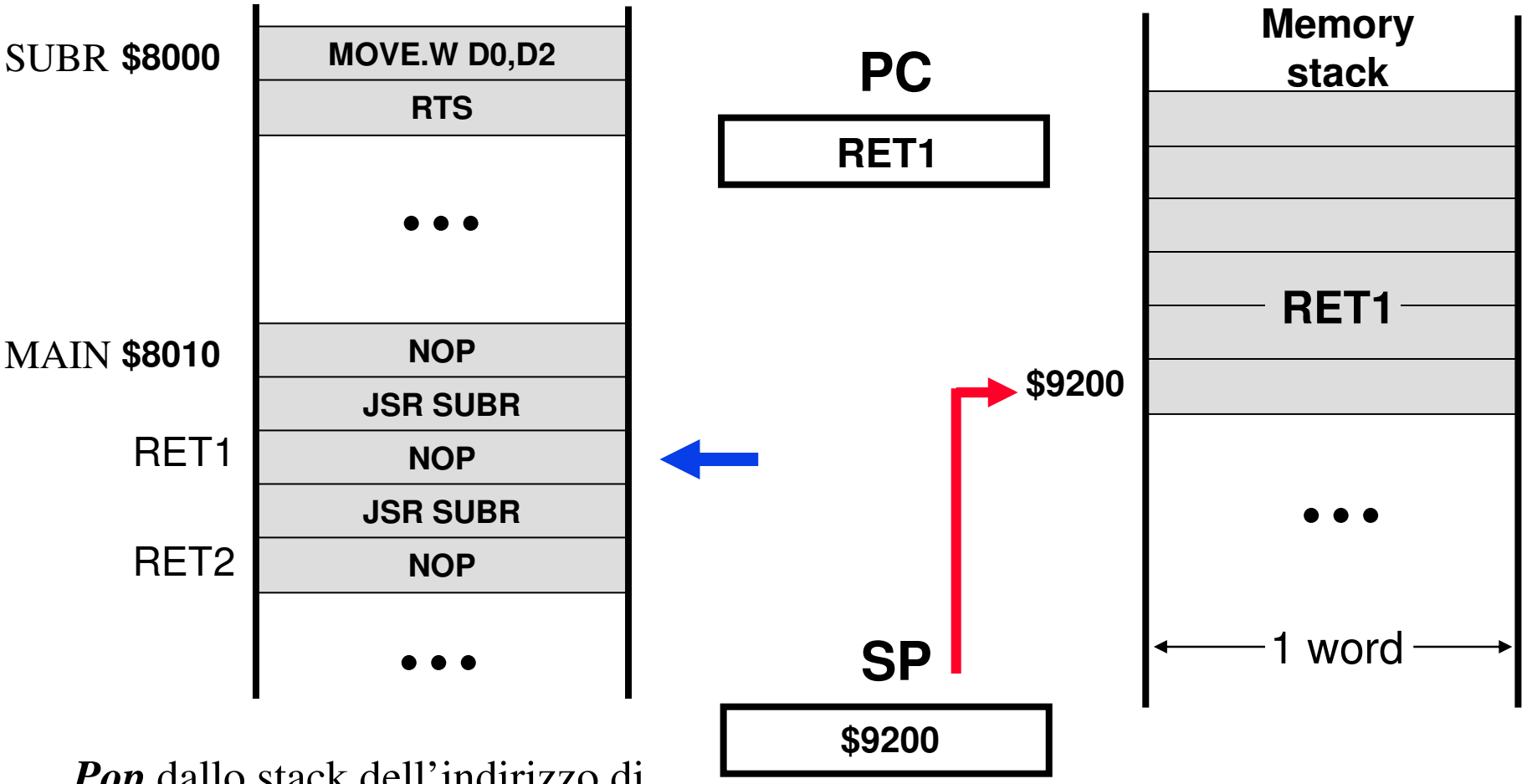
Prima del ritorno al programma chiamante

Esempio - Mappa della memoria e stack



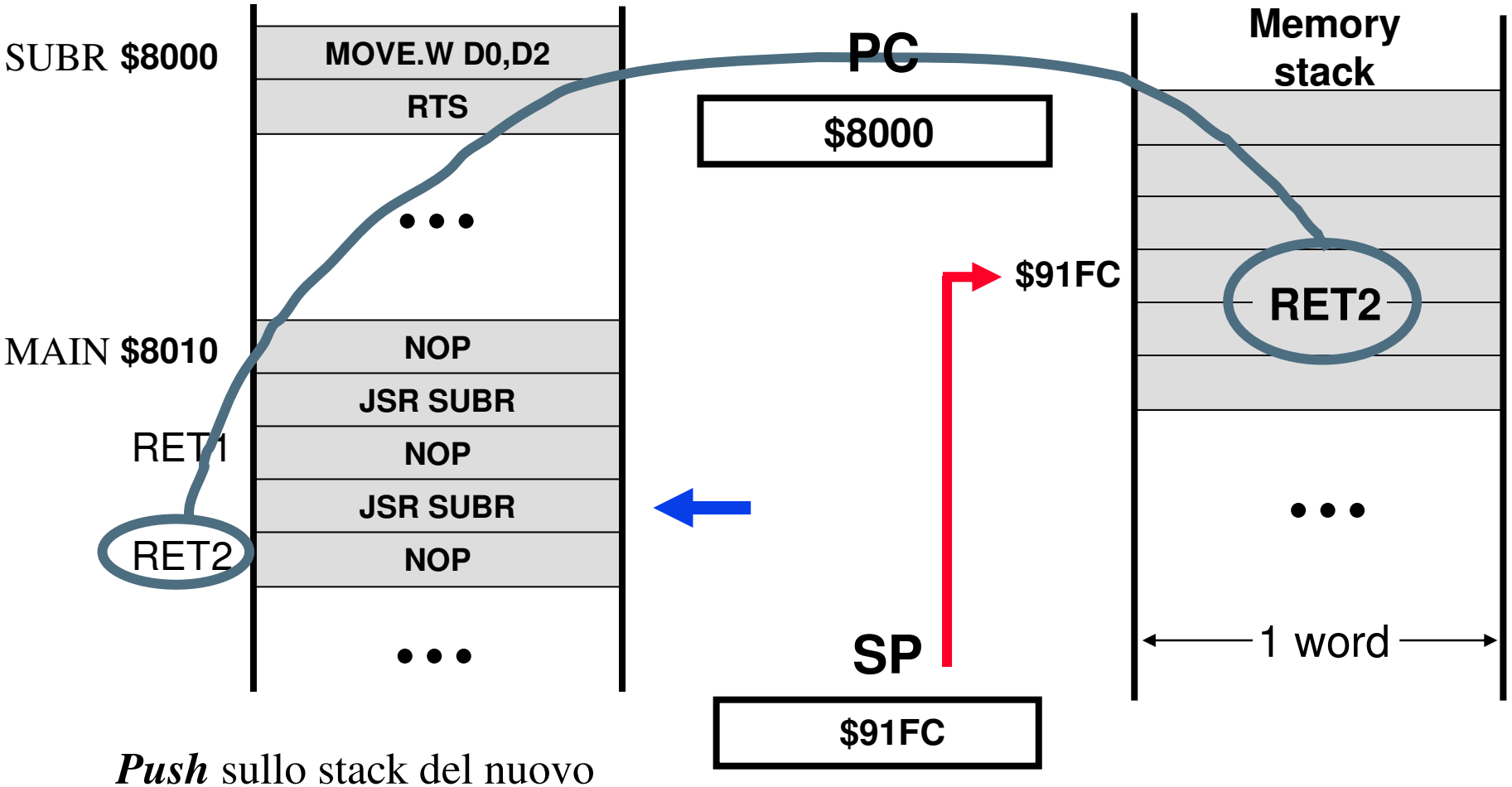
Pop dallo stack dell'indirizzo di ritorno

Esempio - Mappa della memoria e stack



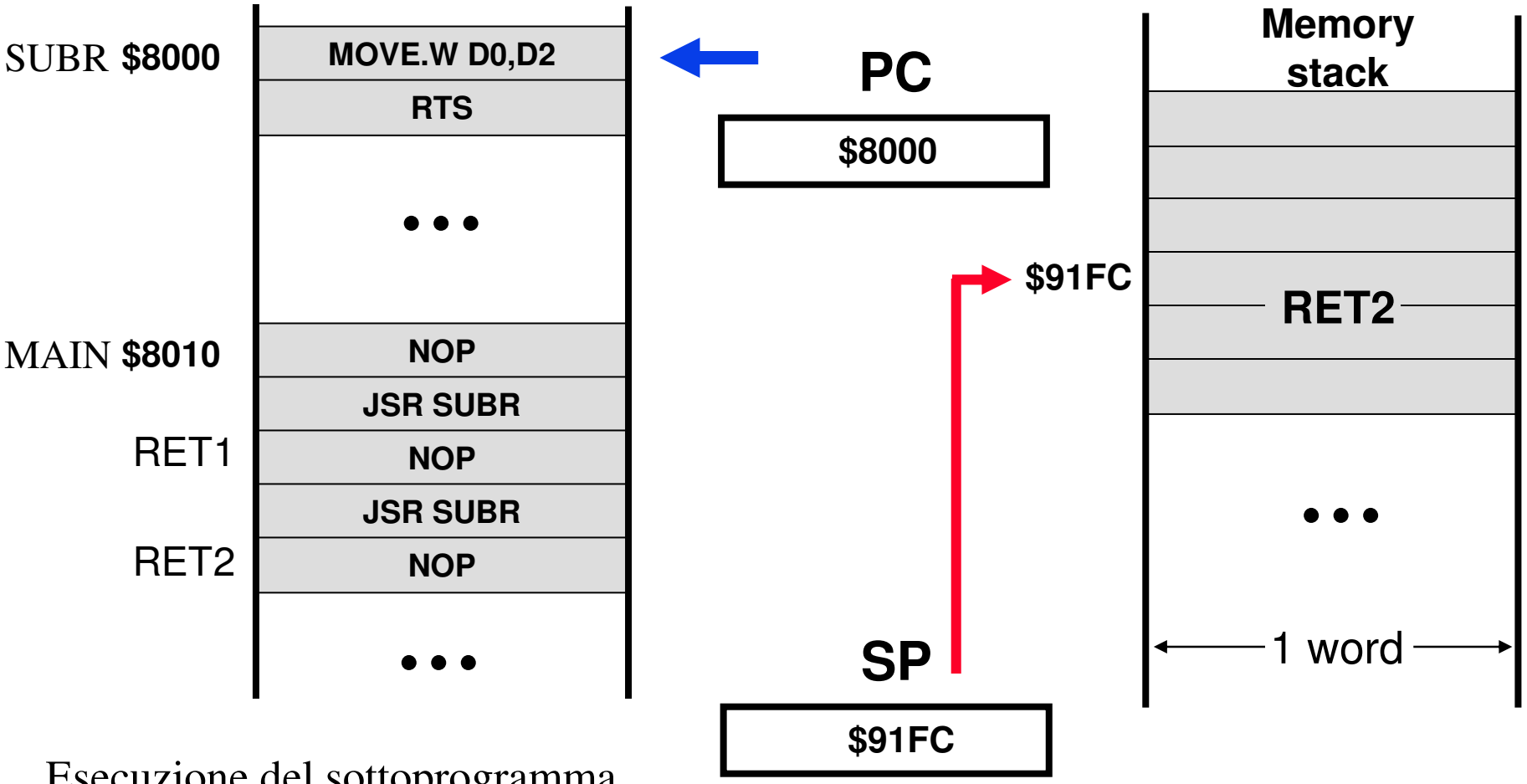
Pop dallo stack dell'indirizzo di ritorno

Esempio - Mappa della memoria e stack



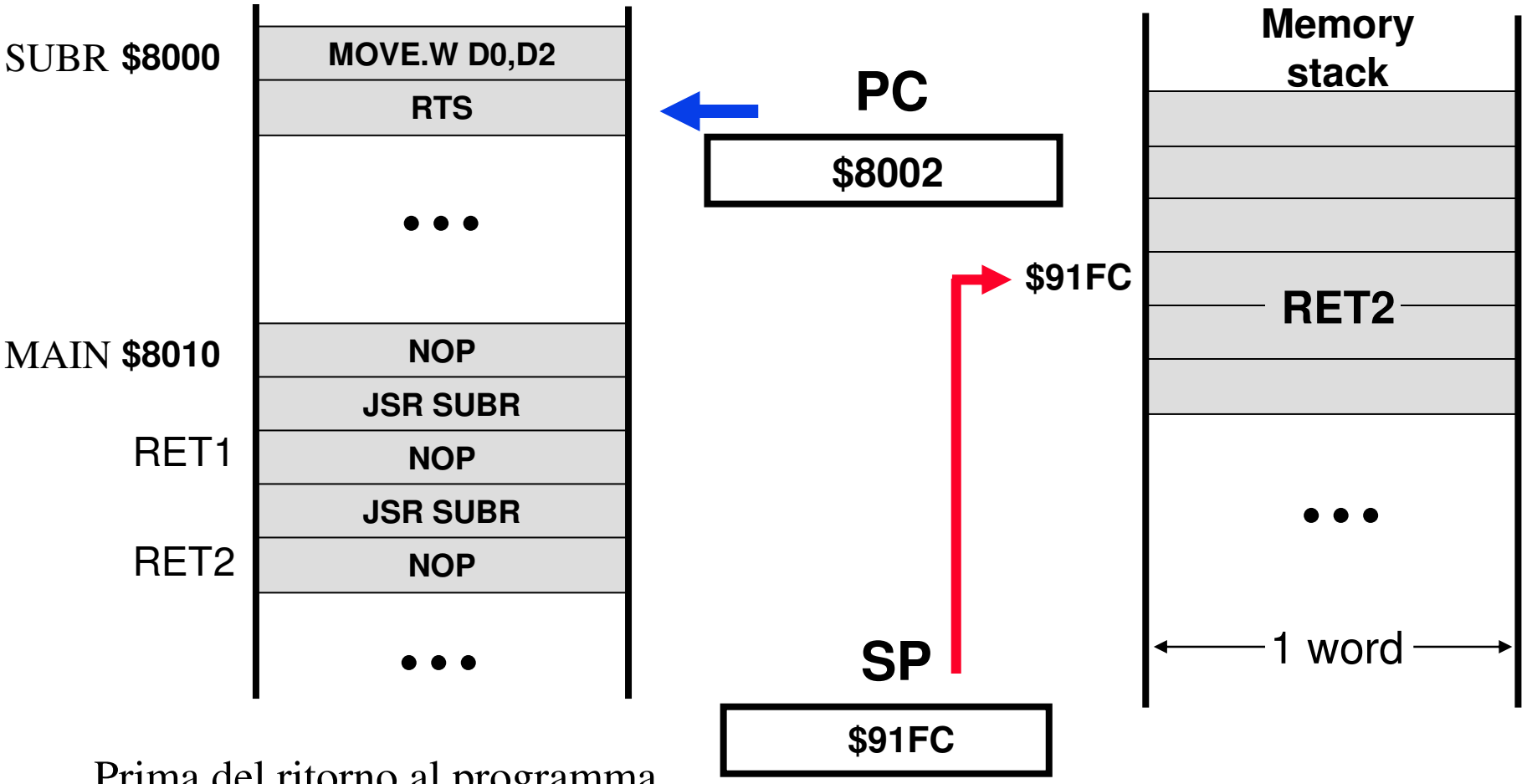
Push sullo stack del nuovo indirizzo di ritorno

Esempio - Mappa della memoria e stack



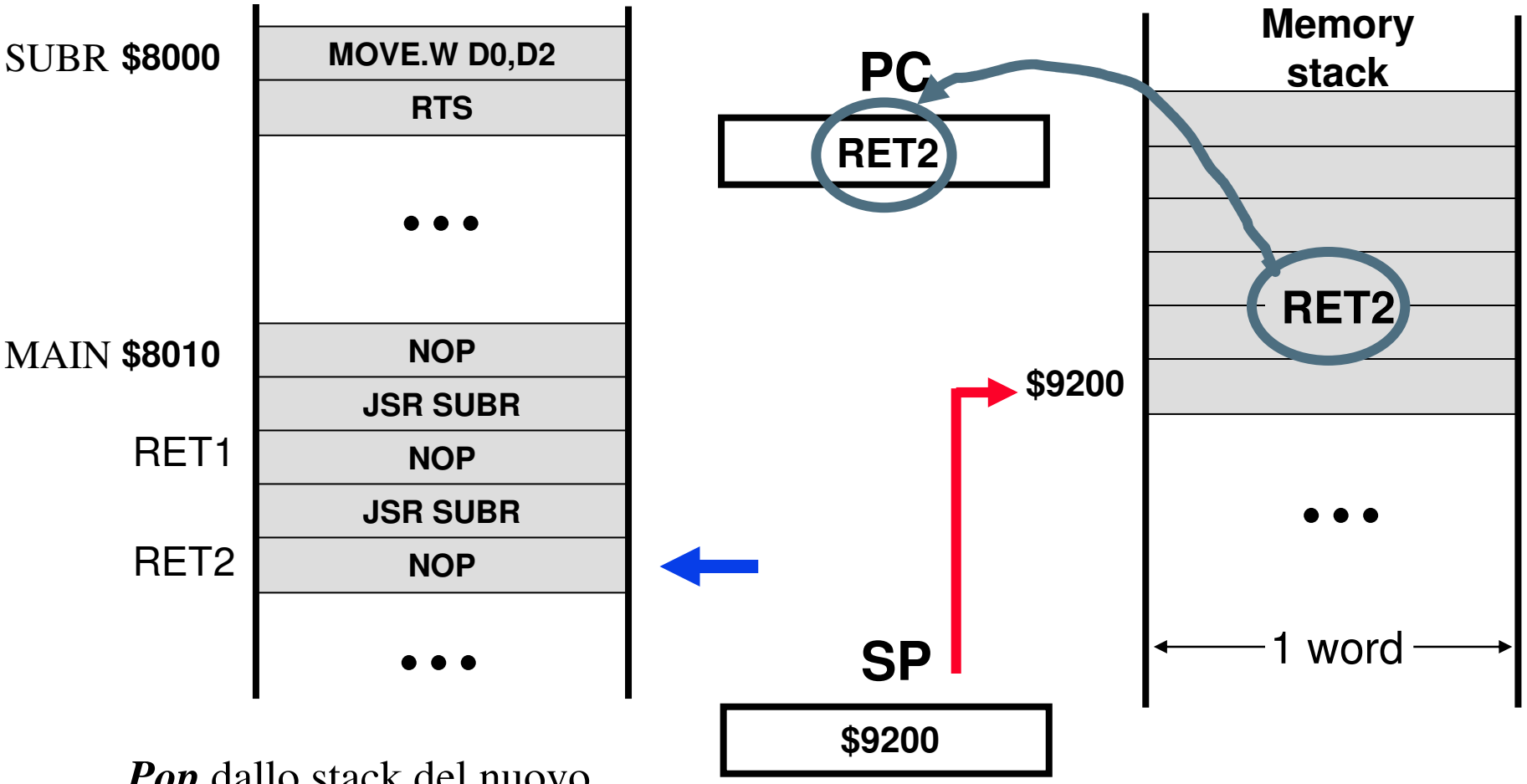
Esecuzione del sottoprogramma

Esempio - Mappa della memoria e stack



Prima del ritorno al programma chiamante

Esempio - Mappa della memoria e stack



Pop dallo stack del nuovo indirizzo di ritorno

Chiamata a Sottoprogramma

- **Nel momento della chiamata a sottoprogramma**, l'istruzione JSR SUBR memorizza l'indirizzo di ritorno sullo stack così:
- il PC che indica l'istruzione successiva viene posto sopra allo stack
- lo SP viene decrementato di 4. Decrementato perchè lo stack cresce verso gli indirizzi piccoli, 4 perchè è la dimensione dell'indirizzo, cioè del registro PC.
- il PC assume il valore dell'indirizzo SUBR, che è la locazione della prossima istruzione da eseguire. Sarà perciò eseguita la prima istruzione del sottoprogramma SUBR.
- All'inizio dell'esecuzione del sottoprogramma, in cima allo stack (cioè all'indirizzo più piccolo dello stack, puntato da SP) troviamo il valore dell'istruzione che segue la JSR del chiamante.

Chiamata a Sottoprogramma

- **Il momento del ritorno da sottoprogramma**, è quello in cui viene eseguita l'istruzione RTS, che:
- carica dalla cima dello stack il valore da assegnare al Program Counter,
- incrementa di 4 il valore dello Stack pointer, per riportare lo stack come prima della chiamata.
- E' importante notare che se il sottoprogramma ha modificato SP, prima della RTS deve rimetterlo al valore corretto, altrimenti la RTS carica nel PC un valore sbagliato ed effettua il ritorno ad un indirizzo errato.

Passaggio dei Parametri al Sottoprogramma

- I dati su cui il sottoprogramma deve lavorare possono essere passati secondo diverse modalità:
- mediante i registri,
- mediante aree dati in memoria,
- mediante lo stack di sistema.
- Il punto fondamentale comunque è sempre l'accordo (convenzione) tra chiamante e chiamato su come effettuare il passaggio.

Passaggio dei parametri tramite Registri.

- Si usa perché è veloce, non serve accedere alla memoria per avere i parametri. Bisogna tenere conto però di quali registri il sottoprogramma usa e cambia.
- Esiste un'istruzione che permette di salvare un'insieme di registri
- MOVEM A0/D2-D4/D6 ,Address
- MOVEM Address, A0/D2-D4/D6
- in particolare per usare lo stack si può chiamare così:
- MOVEM A0/D2-D4/D6 ,-(SP) decrementa lo SP
- MOVEM (SP)+ , A0/D2-D4/D6 incrementa lo SP
- Nella MOVEM è essenziale che l'ordine nella lista nel salvataggio e nel
- ripristino sia lo stesso:
- **MOVEM D0/D1, -(SP)**
- corpo del sottoprogramma
- **MOVEM (SP)+,D0/D1**

Passaggio dei parametri tramite Registri. Esempio

```
*Conta il numero di bit '1' in un byte facendo uso di una subroutine
                    org $8200
BEGIN   movem.l         D0/D1,-(SP)
        move.b          byte,D0
        jsr             n1byte
        move.b          D1,n1word
        movem.l         (SP)+,D0/D1
        stop            #$2700
*Aspetta il byte in D0, fornisce il numero di '1' in D1
n1byte  movem.l A0/D2,-(SP)
        lea masks,A0
        clr.b D1
iciclo  move.b (A0)+,D2
        and.b D0,D2
                        ……..
```

Passaggio dei parametri tramite Registri. Esempio

```
            beq       continua
            add.b     #1,D1
continua    cmp.w     #maske,A0
            bne       iciclo
            movem.l   (SP)+,A0/D2
            rts
                      org $9000
Byte                  dc.b       $1F
n1word                dc.b       0
masks                 dc.b       $80,$40,$20,$10,$8,$4,$2,$1
maske                 equ        masks+8
            end       BEGIN
```

Passaggio di Parametri per Valore o Indirizzo

- L'assembler permette di passare un dato per valore (copia) o per indirizzo, nel secondo caso rendendo modificabile il dato stesso dall'interno di una subroutine.
- Ovviamente nei due casi dovrà essere diverso il modo di accedere al dato.

Passaggio per valore (mediante registro). Risultato restituito mediante un registro

```
            ORG $8000
Dato:       DC.W    $10
SommaV      ADD     #14,D1
            RTS
Start:      MOVE    Dato,D1
            JSR     SommaV
            END     Start
```

Passaggio per indirizzo (mediante registro). Risultato scritto in memoria

```
   ORG     $8000
Dato:      DC.W      $10
Sommal:    ADD       #14,(A1)
           RTS
Start:     LEA       Dato,A1
           JSR       Sommal
           END       Start
```

Esempio

```
                org $8200
MEM1            DC.B        $AA
                org $8300
MEM2            DC.B        $FF
                org $8500
MAIN            MOVE.B      MEM1,D0
                MOVE.B      MEM2,D1
                JSR         SWAP_VAL
                MOVE.L      #MEM1,A0
                MOVE.L      #MEM2,A1
                JSR         SWAP_VAR
                STOP #$2700
*SUBROUTINE SWAP CON SCAMBIO PER VALORE
                org $8500
SWAP_VAL        MOVE.B      D0,D2
                MOVE.B      D1,D0
                MOVE.B      D2,D1
                RTS
*SUBROUTINE SWAP CON SCAMBIO PER INDIRIZZO
                org $8600
SWAP_VAR        MOVE.B      (A0),D0
                MOVE.B      (A1),(A0)
                MOVE.B      D0,(A1)
                RTS
                END         MAIN
```