Lezione 9: Chiamate di sistema con l'istruzione TRAP e chiamate a procedura utente

Elementi di Architettura e Sistemi Operativi
Docente: Tiziano Villa

Corso di Laurea in Bioinformatica

Dicembre 2017

Argomenti della lezione

- Chiamate di sistema mediante l'istruzione TRAP
- Chiamate a procedura utente
- Esempi

Fonte:
Patt & Patel: "*Introduction to Computing Systems: From Bits and Gates to C and Beyond*". Ch. 9.

Chiamate di sistema

Alcune operazioni richiedono competenze e protezioni speciali:

- serve una conoscenza specifica dei registri d'ingresso e uscita e dei modi di usarli;
- bisogna proteggere le risorse d'ingresso e uscita condivise tra piu' utenti.

Sono disponibili delle **procedure di servizio** che si possono invocare con **chiamate di sistema** (system calls, eseguono come codice del sistema operativo) per eseguire operazioni di basso livello che richiedono privilegi.

Chiamate di sistema

- Un programma utente invoca una chiamata di sistema.
- Essa e' eseguita come codice del sistema operativo.
- Alla fine il controllo e' restituito al programma utente.

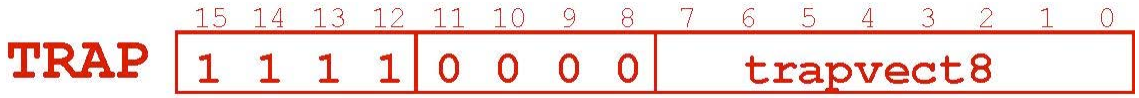
Nell'architettura di LC-3 questo avviene mediante l'istruzione TRAP.

L'istruzione TRAP

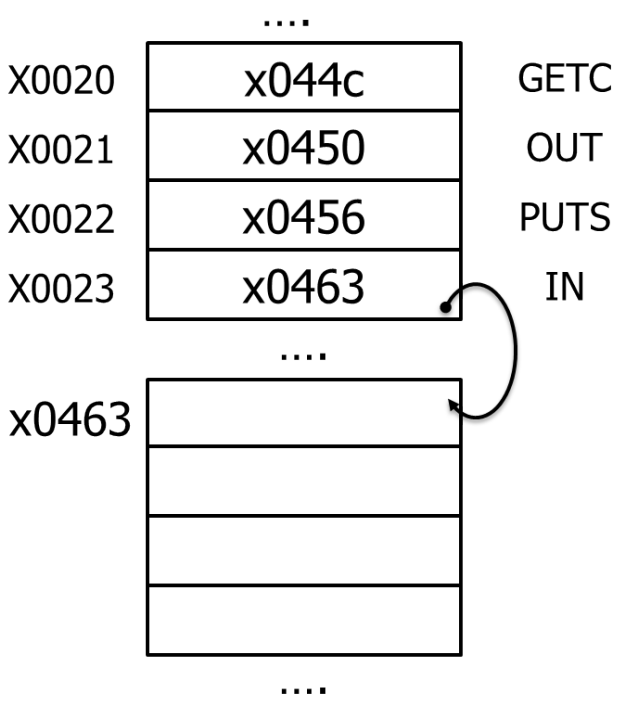
Il meccanismo dell'istruzione TRAP richiede:

- Un insieme di 256 procedure di servizio come parte del sistema operativo (per convenzione il codice di sistema si trova in memoria agl'indirizzi prima di x3000).
- Una tavola degl'indirizzi iniziali in memoria agl'indirizzi da x0000 a x00FF.
- Il trasferimento del controllo al sistema operativo (il vettore trapvect8 - lungo 8 - punta a una delle 256 chiamate di servizio nella tavola precedente).
- Un collegamento (linkage) per tornare al programma chiamante (l'esecuzione prosegue alla riga successiva alla TRAP).

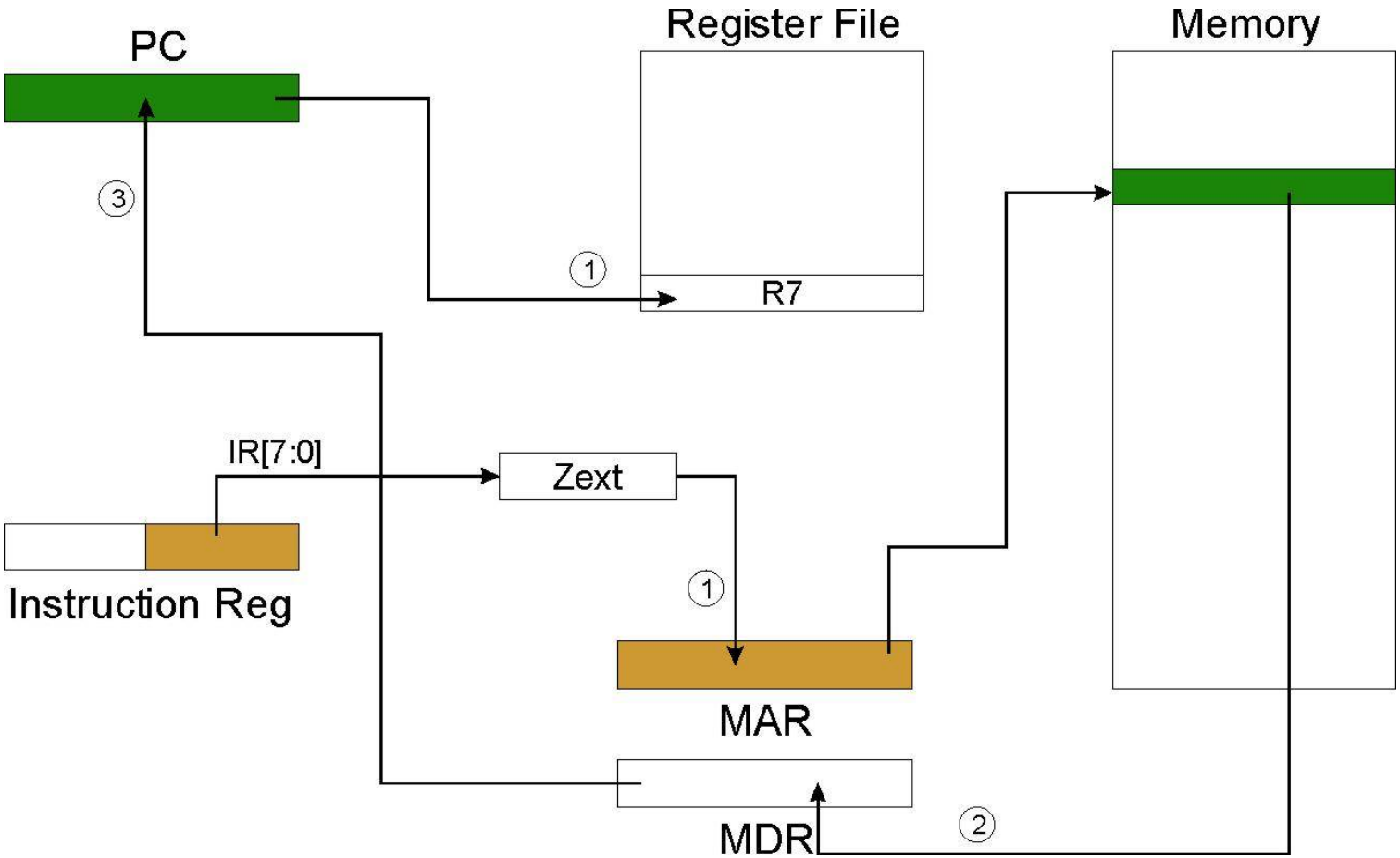
L'istruzione TRAP



- Il vettore `trapvect8` (con zero-estensione) punta a un elemento della tavola degl'indirizzi iniziali delle chiamate di sistema.
- L'istruzione TRAP esegue la procedura di servizio:
  - R7 $\leftarrow$ PC (contatore corrente, indirizzo dell'istruzione successiva);
  - PC $\leftarrow$ indirizzo dell'istruzione iniziale della procedura di servizio.



Esempio di uso dell'istruzione TRAP

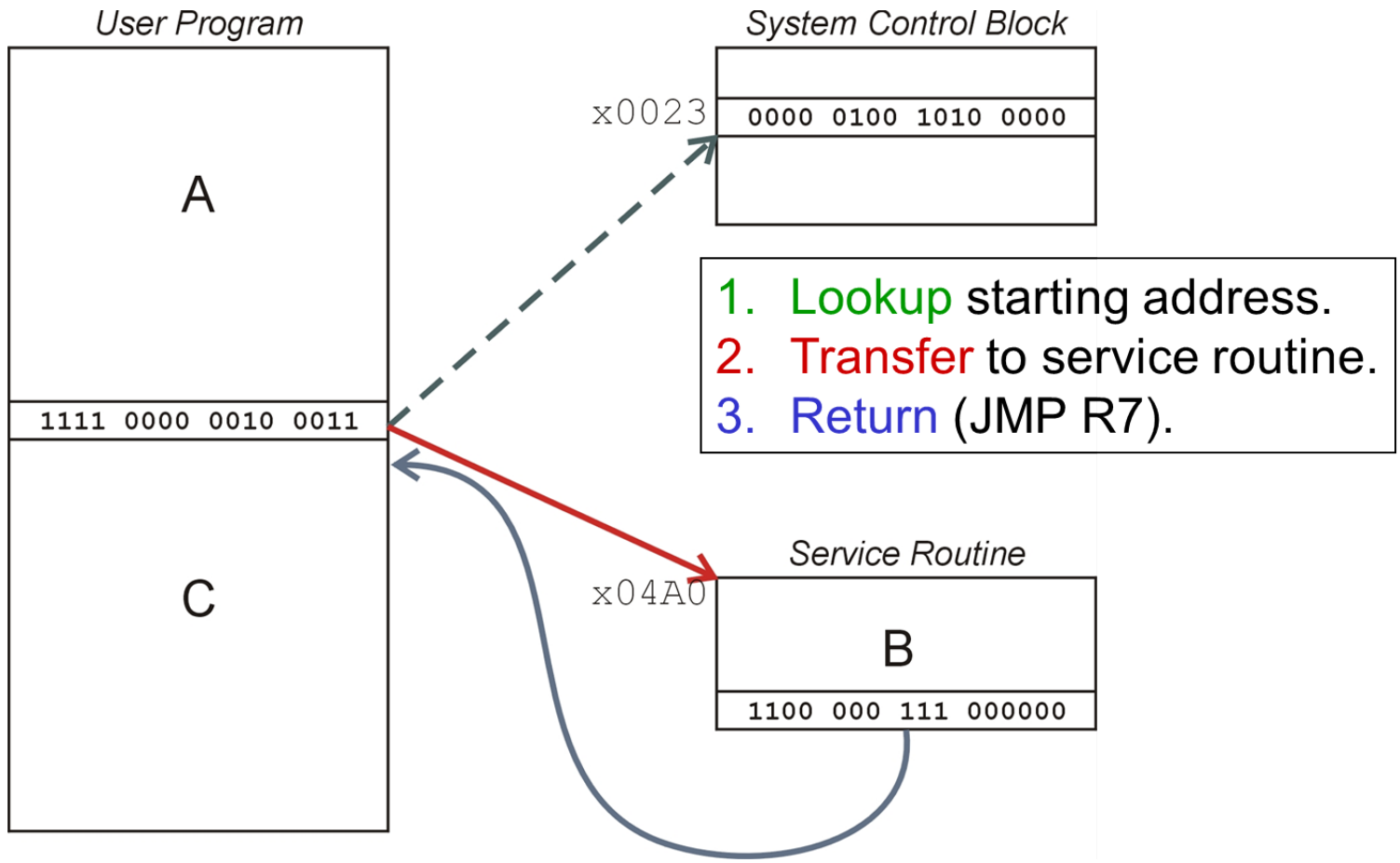


L'istruzione RET (JMP R7)

- Come si trasferisce il controllo all'istruzione dopo la TRAP ?
- Il vecchio contatore di programma e' stato salvato in R7:
  - JMP R7 ci fa ritornare all'istruzione giusta.
  - Il linguaggio macchina di LC-3 definisce RET (return) come equivalente a JMP R7.
- Dobbiamo essere sicuri che la procedura di sistema non modifichi R7, altrimenti perderemo il valore dell'indirizzo di ritorno.

Esempio di esecuzione dell'istruzione TRAP

L'istruzione RET (1100 000 111 000000) si usa per restituire il controllo al programma chiamante.



Procedure di sistema disponibili in LC-3

- GETC (TRAP x20) - Legge un carattere da tastiera nel registro R0[7:0] (azzera R0[15:8]); si trova all'indirizzo x0400
- OUT (TRAP x21) - Scrive dal registro R0[7:0] un carattere sullo schermo; si trova all'indirizzo x0430
- PUTS (TRAP x22) - Scrive sullo schermo una stringa il cui indirizzo del primo carattere si trova nel registro R0; si trova all'indirizzo x0450
- IN (TRAP x23) - Scrive un pronto sullo schermo e legge un carattera da tastiera nel registro R0[7:0] (azzera R0[15:8]) e ne fa eco sullo schermo; si trova all'indirizzo x04A0
- HALT (TRAP x25) - Scrive un messaggio sullo schermo e termina l'esecuzione (ferma l'orologio); si trova all'indirizzo xFD70

Esempio di uso dell'istruzione TRAP

```
            .ORIG x3000
            LD    R2, TERM    ; Load negative ASCII '7'
            LD    R3, ASCII   ; Load ASCII difference
AGAIN       TRAP  x23         ; input character
            ADD   R1, R2, R0  ; Test for terminate
            BRz   EXIT        ; Exit if done
            ADD   R0, R0, R3  ; Change to lowercase
            TRAP  x21         ; Output to monitor...
            BRnzp AGAIN       ; ... again and again...
TERM        .FILL xFFC9       ; -'7'
ASCII       .FILL x0020       ; lowercase bit
EXIT        TRAP  x25         ; halt
            .END
```

Esempio di uso dell'istruzione TRAP

Comportamento del programma precedente:

- L'utente scrive da tastiera una lettera maiuscola e il programma stampa su schermo la lettera minuscola corrispondente.
- Il programma termina quando l'utente scrive un 7 da tastiera.
- Si assume che l'utente scriva da tastiera solo maiuscole o il numero 7.

Procedura di sistema HALT

- Nell'architettura di LC-3 c'e' un lucchetto (latch) RUN che in congiunzione con l'oscillatore produce il segnale dell'orologio che controlla la macchina. Se si azzera tale lucchetto, il segnale si azzera e l'orologio si ferma.
- Il lucchetto RUN e' il campo 15 del registro MCR (Machine Control Register) che e' mappato all'indirizzo di memoria xFFFE.
- Nel seguito si mostra il codice della procedura di sistema che esegue tale operazione.
  - Si salvano i registri R7 (sovrascritto da TRAP x21) , R1 e R0 (usati dalla procedura).
  - Si produce sullo schermo la scritta "Halting the machine".
  - Si azzera il lucchetto RUN mediante una AND del registro MCR con la stringa 0111111111111111.

Procedura di sistema HALT

Domanda: quale istruzione o procedura di servizio si puo' usare per far ripartire l'orologio ?

```
01              .ORIG   XFD70           ; System call starting address
02              ST      R0, SaveR0      ; Saves registers affected
03              ST      R1, SaveR1      ; by routine
04              ST      R7, SaveR7      ;
05
06      ; Print message that machine is halting
07              LD      R0, ASCIINewLine
08              TRAP    x21             ; Set cursor to new line
09              LEA     R0, Message     ; Get start of message
0A              TRAP    x22             ; and write it to monitor
0B              LD      R0, ASCIINewLine
0C              TRAP    x21
0D
0E      ; Clear MCR[15] to stop the clock
0F              LDI     R1, MCR         ; Load MC register to R1
10              LD      R0, MASK        ; MASK = x7FFF (i.e. bit 15 = 0)
11              AND     R0, R1, R0      ; Clear bit 15 of copy of MCR
12              STI     R0, MCR         ; and load it back to MCR
```

Procedura di sistema HALT

Domanda: come puo' la procedura eseguire le istruzioni indicate dopo l'esecuzione della STI che ferma l'orologio ?

```
13      ; Return from the HALT routine
14      ; (how can this ever happen, if the clock is stopped on line 12??)
15      ;
16                      LD      R7, SaveR7      ; Restores registers
17                      LD      R1, SaveR1      ; before returning
18                      LD      R0, SaveR0
19                      RET                     ; JMP R7
1A
1B      ; constants
1C      ASCIINewLine    .FILL   x000A
1D      SaveR0          .BLKW   1
1E      SaveR1          .BLKW   1
1F      SaveR7          .BLKW   1
20      Message         .STRINGZ    "Halting the machine"
21      MCR             .FILL   xFFFE
22      MASK            .FILL   x7FFF
23                      .END
```

Procedura di sistema HALT

- Solo un meccanismo esterno (**un tasto esterno di ripartenza pigiato manualmente**) puo' far ripartire l'orologio dopo che e' stato fermato. Dopo l'esecuzione dell'istruzione STI R0, MCR della procedura HALT la macchina si ferma (si e' azzerato MCR[15] e quindi si ferma l'orologio) e percio' non si puo' piu' eseguire alcuna istruzione.
- Quando la macchina riparte, la procedura HALT riprendera' da LD R1, SaveR1.
- Alla fine dell'esecuzione di HALT, il contatore di programma puntera' all'indirizzo dell'istruzione che segue la chiamata di HALT.
- Possibile uso di HALT: fermare la macchina ad es. durante un programma di manutenzione, esaminarne lo stato, poi farla ripartire con un tasto esterno e tornare al programma di manutenzione.

Salvataggio e ripristino dei registri

- Si deve salvare il valore di un registro se:
  - Il suo valore sara' sovrascritto dalla procedura di servizio.
  - Ci servira' usare tale valore nuovamente.
- Chi deve salvare ?
  - Il chiamante ? Sa che cosa serve dopo, ma non sa che cosa sara' sovrascritto dal chiamato.
  - Il chiamato ? Sa che cosa sovrascrive, ma non sa che cosa servira' di nuovo al chiamante.

Esempio di salvataggio e ripristino dei registri

S'inseriscono da tastiera 10 cifre decimali, si convertono i loro codici ASCII in binario, e si memorizzano i valori binari in 10 locazioni consecutive a partire dall'indirizzo `Binary`.

```
            LEA   R3, Binary
            LD    R6, ASCII   ; char->digit template
            LD    R7, COUNT   ; initialize to 10
AGAIN       TRAP x23          ; Get char
            ADD   R0, R0, R6  ; convert to number
            STR   R0, R3, #0  ; store number
            ADD   R3, R3, #1  ; incr pointer
            ADD   R7, R7, -1  ; decr counter
            BRp   AGAIN       ; more?
            BRnzp NEXT
ASCII       .FILL     xFFD0
COUNT       .FILL     #10
Binary      .BLKW #10
```

Esempio di salvataggio e ripristino dei registri

Problema con la realizzazione precedente: l'istruzione TRAP (riga 04) rimpiazza il valore 10 (caricato in R7 alla riga precedente) con l'indirizzo dell'istruzione ADD R0,R0,R6. Percio' le istruzioni alle righe 08 e 09 (ADD R7,R7,-#1 e BRp AGAIN) non funzionano correttamente nel ciclo in cui sono inserite.
Per risolvere il problema: salvare in memoria il registro R7 prima della TRAP e ripristinarlo dopo la TRAP.

Salvataggio e ripristino dei registri

- Procedura chiamata:
  - All'inizio: salvare i registri che saranno usati (a meno che i valori modificati servano al chiamante). Es. la procedura chiamata HALT salva all'inizio i registri R1 e R0 (istruzioni ST).
  - Alla fine: ripristinare tali registri. Es. la procedura chiamata HALT ripristina alla fine i registri R1 e R0 (istruzioni LD).
- Procedura chiamante:
  - Salvare i registri modificati dalle proprie istruzioni o dalle procedure chiamate (se noto), se i loro valori servono dopo. Esempi:
    - HALT come procedura chiamante salva R7 prima della prima chiamata di TRAP e lo ripristina dopo l'ultima chiamata di TRAP.
    - La procedura chiamante salva R0 prima di chiamare TRAP x023 (legge un carattere da tastiera con eco sullo schermo).
  - In alternativa evita di usare tali registri.

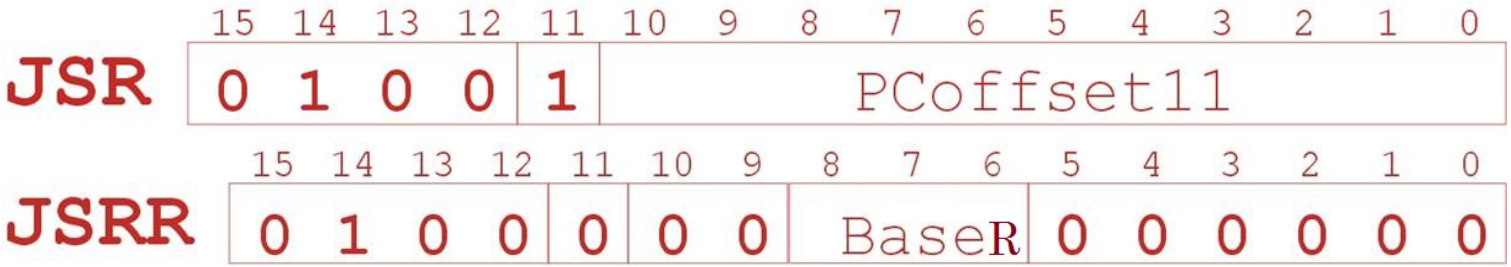
Chiamate a procedure utente

Una procedura (funzione, subroutine) utente e' un'unita' di codice richiamabile. E' utile per riuso di codice di utente o libreria, e la modularizzazione di compiti complessi.

A differenza di una procedura di servizio non e' parte del sistema operativo, perche' non ha privilegi speciali e non deve proteggere risorse di sistema

L'architettura di LC-3 mette a disposizione l'istruzione JSR(R) (Jump to Subroutine) per chiamate a procedura.

Semantica dell'istruzione JSR(R)

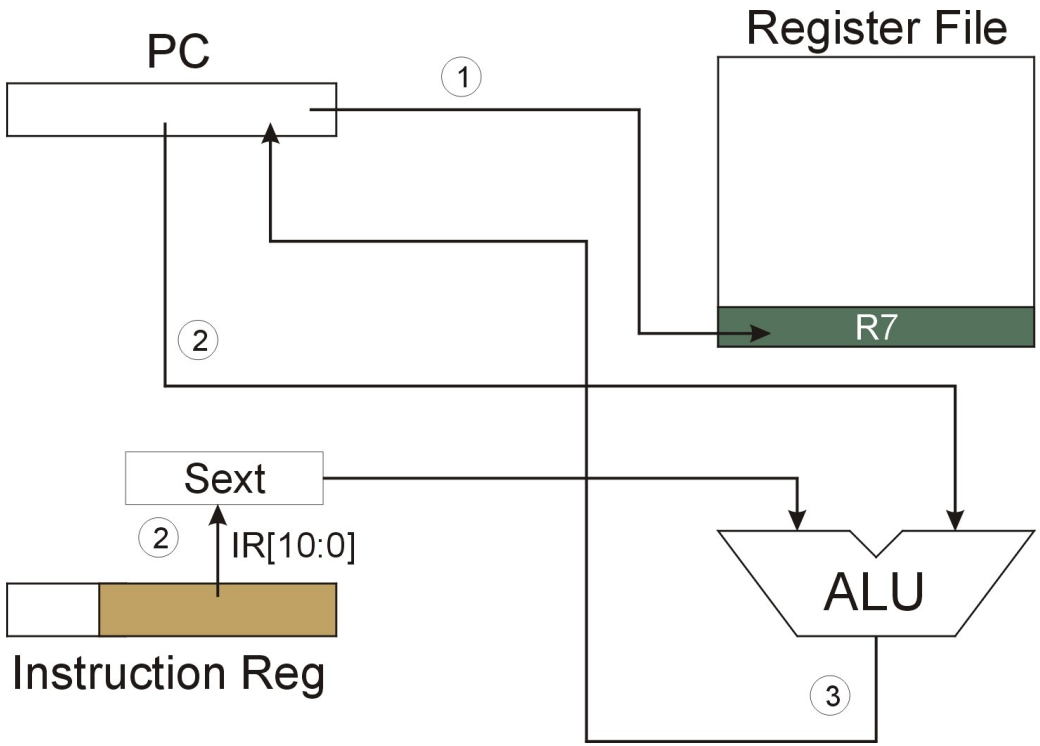


- Salva il contatore di programma in R7 (indirizzo della prossima istruzione) e salta incondizionatamente a una locazione.
  - Il salvataggio dell'indirizzo di ritorno si chiama "collegamento" (linking).
  - Il campo IR[11] specifica il modo d'indirizzamento:
    - ★ IR[11]=1, salta a indirizzo = PC+Sext(IR[10:0]),
    - ★ IR[11]=0, salta a indirizzo = contenuto registro base IR[8:6].

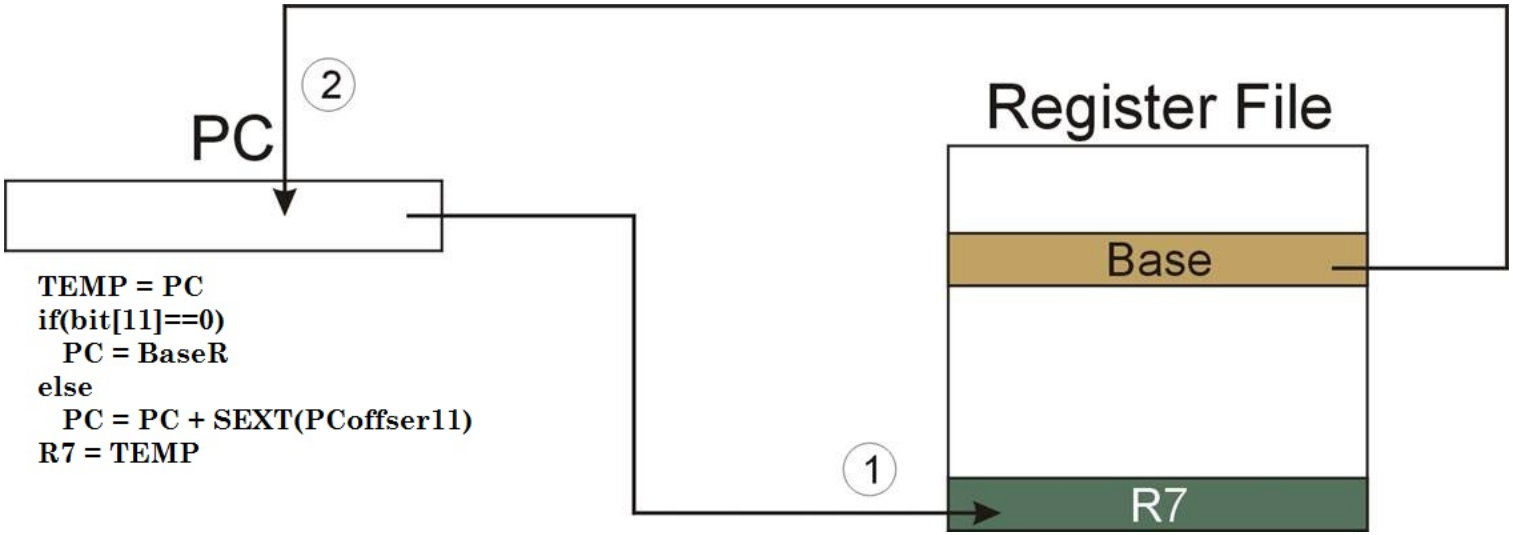
Semantica dell'istruzione JSR(R)

- Sia la seguente istruzione JSR 0100 1 10000000100 memorizzata all'indirizzo x4200 (PC), dalla sua esecuzione si ha $R7 \leftarrow x4201(PC + 1)$ e $PC \leftarrow x3E05(PC + 1 + Ext(PCoffset11) = x4201 + xFC04)$.
- Sia la seguente istruzione JSRR 0100 0 00 101 000000 memorizzata all'indirizzo x420A (PC) e R5 contenga x3002, dalla sua esecuzione si ha $R7 \leftarrow x420B(PC + 1)$ e $PC \leftarrow x3002(R5)$.

Esecuzione dell'istruzione JSR



Esecuzione dell'istruzione JSRR



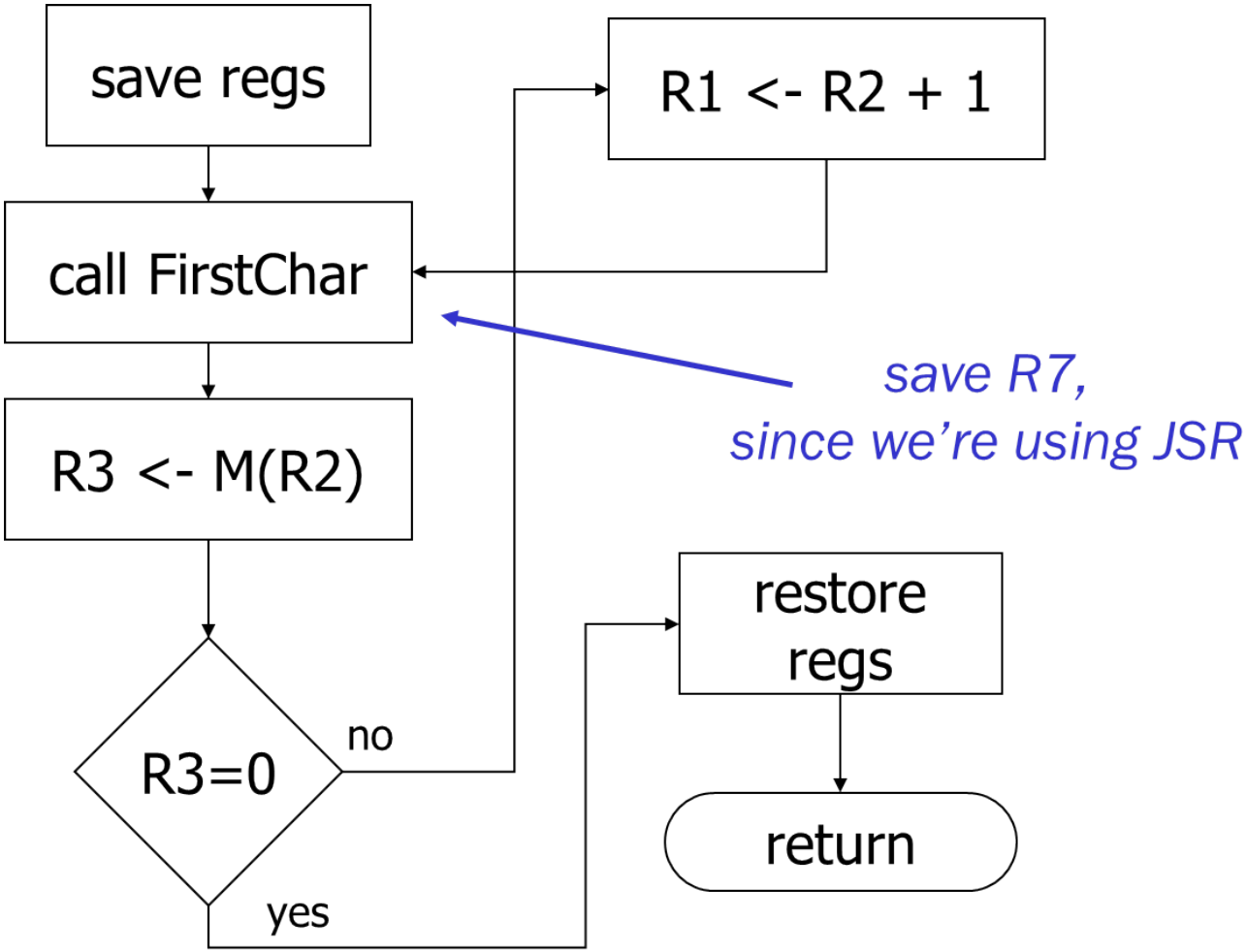
Passaggio d'informazioni con le procedure

- Per usare una procedura, il programmatore deve conoscere:
  - il suo indirizzo (almeno come etichetta simbolica);
  - la sua funzione;
  - i suoi argomenti (i dati passati alla procedura);
  - i suoi valori di ritorno (i dati restituiti dalla procedura).
- Una convenzione e' che il chiamato salvi tutti i registri che saranno modificati internamente (non visibili al ritorno) e i valori degli argomenti (se non sovrascritti dai valori di ritorno).
- Il chiamato deve salvare R7 se chiama una qualsiasi altra procedura d'utente o di servizio (cioe' e' anche chiamante), altrimenti non potra' tornare al chiamante.

Esempio

- Si scriva una procedura FirstChar per trovare la prima occorrenza in una stringa (a cui punta R1) di un particolare carattere (che si trova in R0); si restituisca come risultato in R2 il puntatore al carattere o a fine stringa (NULL).
- Si utilizzi la procedura FirstChar per scrivere una procedura CountChar, che conta il numero di occorrenze in una stringa (a cui punta R1) di un particolare carattere (che si trova in R0); si restituisca come risultato in R2 tale conteggio.

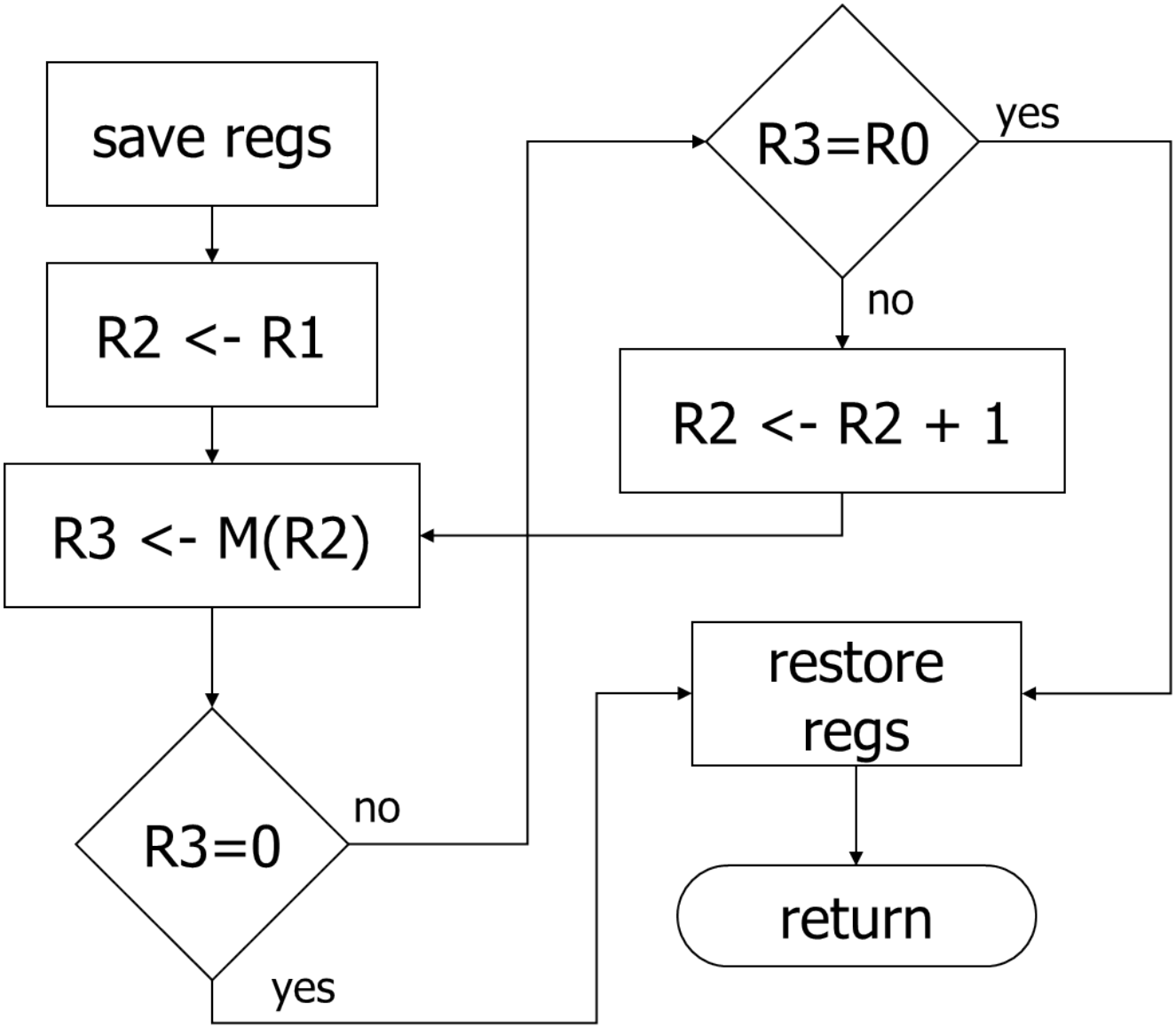
Algoritmo della procedura CountChar (usa FirstChar)



Realizzazione della procedura CountChar

```
; CountChar: subroutine to count occurrences of a char
CountChar
        ST     R3, CCR3      ; save registers
        ST     R4, CCR4
        ST     R7, CCR7      ; JSR alters R7
        ST     R1, CCR1      ; save original string ptr
        AND    R4, R4, #0    ; initialize count to zero
CC1     JSR    FirstChar     ; find next occurrence (ptr in R2)
        LDR    R3, R2, #0    ; see if char or null
        BRz    CC2           ; if null, no more chars
        ADD    R4, R4, #1    ; increment count
        ADD    R1, R2, #1    ; point to next char in string
        BRnzp  CC1
CC2     ADD    R2, R4, #0    ; move return val (count) to R2
        LD     R3, CCR3      ; restore regs
        LD     R4, CCR4
        LD     R1, CCR1
        LD     R7, CCR7
        RET                  ; and return
```

Algoritmo della procedura FirstChar



Realizzazione della procedura FirstChar

```
; FirstChar: subroutine to find first occurrence of a char
FirstChar
        ST     R3, FCR3      ; save registers
        ST     R4, FCR4      ; save original char
        NOT    R4, R0        ; negate R0 for comparisons
        ADD    R4, R4, #1
        ADD    R2, R1, #0    ; initialize ptr to beginning of string
FC1     LDR    R3, R2, #0    ; read character
        BRz    FC2           ; if null, we're done
        ADD    R3, R3, R4    ; see if matches input char
        BRz    FC2           ; if yes, we're done
        ADD    R2, R2, #1    ; increment pointer
        BRnzp  FC1
FC2     LD     R3, FCR3      ; restore registers
        LD     R4, FCR4      ;
        RET                  ; and return
```

Uso dei registri in CountChar/FirstChar

CountChar:

- R0 - contiene carattere da cercare
- R1 - puntatore al primo carattere della stringa
- R2 - puntatore al carattere restituito da FirstChar
- R3 - carattere puntato da R2 (cioe' contenuto dell'indirizzo in R2)
- R4 - conteggio
- R7 - indirizzo ritorno CountChar

FirstChar:

- R3 - carattere puntato da R2 (cioe' contenuto dell'indirizzo in R2)
- R4 - complemento a due del carattere in R0

Conclusioni

- Le procedure invocate tramite l'istruzione TRAP sono scritte dai programmatori di sistema (di fatto costituiscono un sistema operativo rudimentale):
  - interagiscono con funzionalita' specifiche del sistema;
  - spesso richiedono privilegi di accesso.
- Le procedure ordinarie sono scritte dai programmatori di applicativi.
- Le procedure sono spesso aggregate in librerie, come ad esempio `math`. Le etichette per le procedure in librerie esterne si definiscono con la parola chiave .External seguita dall'etichetta.
- Ogni procedura contiene la sua propria tavola dei simboli, e il programma collegatore (linker) risolve gl'indirizzi esterni prima di creare l'eseguibile.