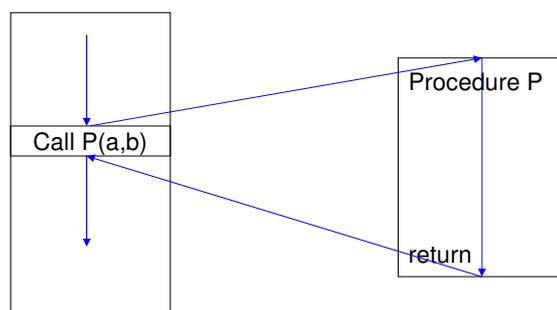


הרצאה 5 שגרות Subroutines

Edited by Tamer Salman 2008

1

שגרות: מה ומדוע?



• מה?

• מדוע?

- חלוקת התוכנית ליחידות מודולריות.
- לא לשכפול קוד.
- תמיכה בפונקציות סטנדרטיות.

Edited by Tamer Salman 2008

2

מנגנונים דרושים

- העברת בקרה
 - קריאה (call)
 - חזרה (return)
- העברת פרמטרים (ארגומנטים)
- הערה
 - יש להימנע מתופעות לוואי (side effects): שגרה לא תשנה שטחים גלובליים (כגון רגיסטרים), אלא רק פרמטרים.

Edited by Tamer Salman 2008

3

מימוש שגרות ב-PDP-11

- קריאה לשגרה
 - JSR Rn, Subr
- חזרה משגרה
 - RTS Rn
- מגבלה
 - Rn ≠ R6 (SP)
- רגיסטר קישור (Linkage Register)
 - Rn הוא רגיסטר הקישור המשמש לרישום כתובת החזרה.
- בתהליך בקרת הקריאה לשגרה משתמשים במחסנית (Stack).

Edited by Tamer Salman 2008

4

JSR Rn, Subr

Push Rn

Rn ← PC

PC ← Subr

- בביצועה שלושה שלבים

– שמירת ערך Rn:

– שמירת כתובת החזרה:

– קפיצה לשגרה:

RTS Rn

PC ← Rn

Pop Rn

- בביצועה שני שלבים

– שחזור כתובת החזרה:

– שחזור ערך Rn:

Edited by Tamer Salman 2008

5

ה-PC כרגיסטר קישור

JSR PC, Subr

Push PC

~~PC ← PC~~

PC ← Subr

– שמירת ערך Rn:

~~– שמירת כתובת החזרה:~~

– קפיצה לשגרה:

RTS PC

~~PC ← PC~~

Pop PC

~~– שחזור כתובת החזרה:~~

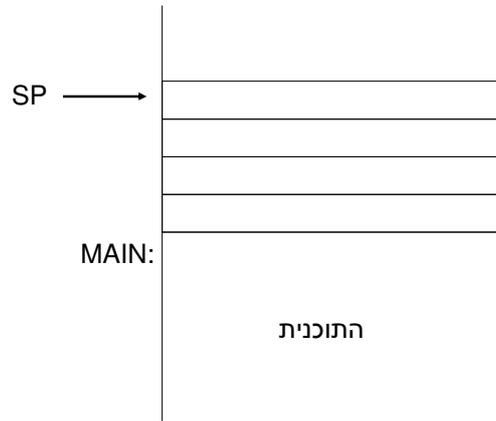
– שחזור ערך Rn:

- עדיף להשתמש ב-PC כרגיסטר קישור, אך לא תמיד זה אפשרי. (תשובה בהעברת פרמטרים בשיטת IN-LINE).

Edited by Tamer Salman 2008

6

המחסנית



Edited by Tamer Salman 2008

7

פעולות על המחסנית

• הגדרה: ה-SP מצביע על האיבר שבראש המחסנית.

• אתחול (init)

```
MOV #MAIN, SP
```

או אתחול אלטרנטיבי

```
MAIN: MOV PC, SP  
TST -(SP)
```

• הכנסת איבר (push)

```
MOV ..., -(SP)
```

• הוצאת איבר (pop)

```
MOV (SP)+, ...
```

• קריאה או כתיבה בעומק המחסנית

```
... Δ(SP) ...
```

Edited by Tamer Salman 2008

8

העברת פרמטרים

- אופן ההעברה
 - לפי ערך (by value)
 - לפי כתובת (by address)
- שטח ההעברה
 - ברגיסטר (in register)
 - בשטח משותף (in common area)
 - בקוד (in line)
 - במחסנית (in stack)

In Register

- התוכנית הקוראת: תכניס את הפרמטר לתוך רגיסטר שהשגרה קבעה לה מראש ותקרא לשגרה.
- השגרה: תשתמש ברגיסטר.
- יתרון: קל ומהיר.
- חסרון: אין הרבה.
- דוגמה:

```
MAIN:                               Subr:
:                                   :
MOV #5, R1                          שימוש ברגיסטר R1
JSR PC, Subr                         :
:
```

In Common Area

- התוכנית הקוראת: תכניס את הפרמטר לתוך משתנה בשטח גלובלי שנקבע מראש ע"י הגשרה.
- השגרה: תשתמש בשטח הגלובלי.
- יתרון: נוח...
- חסרון: לא מודולרי, למי שה שייך? אין הסתרה...
- דוגמה:

```

MAIN:                               Subr:
    :                               :
    MOV #5, Var                     שימוש ב- Var
    JSR PC, Subr                     :
    :                               :
Var: .word 0
    
```

Edited by Tamer Salman 2008

11

In Line

- התוכנית הקוראת: תקרא לשגרה ותרשום מיד לאחר הקריאה את רשימת הפרמטרים.
- השגרה: תקרא אותם בעזרת רגיסטר הקישור ותקדם אותו לפקודה הבאה בתוכנית הקוראת תוך כדי.
- יתרון: רואים את הפרמטרים בקלות מיד אחרי הקריאה.
- חסרון: אין שימוש ב-PC. קריאת הפרמטרים מסובכת. חייבים לקרוא את כולם.
- דוגמה:

```

MAIN:                               Subr:
    :                               :
    JSR R5, Subr                     MOV (R5)+, ...
    .word 3, 5                        MOV (R5)+, ...
    :                               :
    
```

Edited by Tamer Salman 2008

12

In Stack

- התכנית הקוראת: תכניס את הפרמטרים למחסנית ותקרא לשגרה. בסוף היא תנקה אותם מהמחסנית.
- השגרה: תקרא אותם מהמחסנית בעזרת $\Delta(SP)$.
- יתרון: מודולרי. אין הגבלה (כמעט).
- חסרון: ?
- דוגמה:

```

MAIN:
:
MOV #3, -(SP)
MOV #5, -(SP)
JSR R5, Subr
MOV (SP)+, (SP)+
:

Subr:
:
MOV 6(SP) ...
MOV 10(SP), ...
:
    
```

Edited by Tamer Salman 2008

13

מה עדיף?

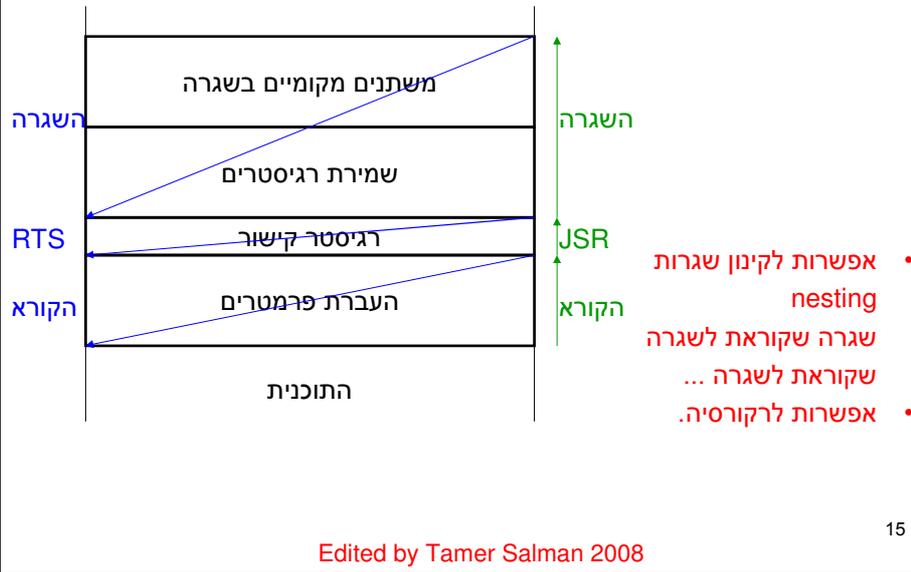
- In Register הוא הכי קל ומהיר.
 - In Stack הוא הכי טוב להרבה פרמטרים.
 - למשל, אם השגרות שלנו יהיו כמו פונקציות בשפת C, כלומר מקבלות מספר כלשהו של פרמטרים ומחזירות אחד בלבד, כגון: `<type1> Function (<type2> p1, <type3> p2, ...)`
- אז נעדיף In Stack עבור פרמטרי הקלט ו-In Register עבור פרמטר הפלט.

הערה: מערך ניתן להעביר אך ורק By Address.

Edited by Tamer Salman 2008

14

תוכן המחסנית בזמן קריאה לשגרה



דוגמה

- Rem10 היא שגרה המחשבת את שארית החלוקה ב-10 של מספר נתון.
- Main היא התוכנית הקוראת לשגרה Rem10.
- לשגרה קלט יחיד ופלט יחיד.

By value – In register

```
Rem10:    mov    r0, -(sp)
          mov    r0, r1
          sxt    r0
          div   #10., r0
          mov   (sp)+, r0
          rts   pc

Main:     ...
          mov   X, r0
          jsr   pc, Rem10
          mov  r1, Y

X:        .word 35.
Y:        .word 0
```

Edited by Tamer Salman 2008

17

By address – In register

```
Rem10:    mov    r0, -(sp)
          mov    (r0), r1
          sxt    r0
          div   #10., r0
          mov   (sp)+, r0
          rts   pc

Main:     ...
          mov   #X, r0
          jsr   pc, Rem10
          mov  r1, Y

X:        .word 35.
Y:        .word 0
```

Edited by Tamer Salman 2008

18

By value – In common area

```
Rem10:    mov    r0, -(sp)
          mov    r1, -(sp)
          mov    Arg, r1
          sxt    r0
          div   #10., r0
          mov    r1, Arg+2
          mov    (sp)+, r1
          mov    (sp)+, r0
          rts    pc
```

```
Main:     ...
          jsr    pc, Rem10
          halt
```

```
Arg:      .word 35.
          .word 0
```

Edited by Tamer Salman 2008

19

By address – In common area

```
Rem10:    mov    r0, -(sp)
          mov    r1, -(sp)
          mov    @Arg, r1
          sxt    r0
          div   #10., r0
          mov    r1, @Arg+2
          mov    (sp)+, r1
          mov    (sp)+, r0
          rts    pc
```

```
Main:     ...
          jsr    pc, Rem10
          halt
```

```
Arg:      .word X
          .word Y
```

```
X:        .word 35.
Y:        .word 0
```

Edited by Tamer Salman 2008

20

By value – In line

```
Rem10:    mov    r0, -(sp)
          mov    r1, -(sp)
          mov    (r5)+, r1
          sxt    r0
          div   #10., r0
          mov    r1, (r5)+
          mov    (sp)+, r1
          mov    (sp)+, r0
          rts   r5

Main:     ...
          jsr   r5, Rem10
          .word 35.
          .word 0
          halt
```

Edited by Tamer Salman 2008

21

By address – In line

```
Rem10:    mov    r0, -(sp)
          mov    r1, -(sp)
          mov    @(r5)+, r1
          sxt    r0
          div   #10., r0
          mov    r1, @(r5)+
          mov    (sp)+, r1
          mov    (sp)+, r0
          rts   r5

Main:     ...
          jsr   r5, Rem10
          .word X
          .word Y
          halt

X:        .word 35.
Y:        .word 0
```

Edited by Tamer Salman 2008

22

By value – In stack

```
Rem10:    mov    r0, -(sp)
          mov    r1, -(sp)
          mov    6(sp), r1
          sxt    r0
          div   #10., r0
          mov    r1, 6(sp)
          mov    (sp)+, r1
          mov    (sp)+, r0
          rts    pc

Main:     ...
          mov    X, -(sp)
          jsr   pc, Rem10
          mov    (sp)+, Y

X:        .word 35.
Y:        .word 0
```

Edited by Tamer Salman 2008

23

By address – In stack

```
Rem10:    mov    r0, -(sp)
          mov    r1, -(sp)
          mov    @6(sp), r1
          sxt    r0
          div   #10., r0
          mov    r1, 6(sp)
          mov    (sp)+, r1
          mov    (sp)+, r0
          rts    pc

Main:     ...
          mov    #X, -(sp)
          jsr   pc, Rem10
          mov    (sp)+, Y

X:        .word 35.
Y:        .word 0
```

Edited by Tamer Salman 2008

24