v.18-04-2024

ΔΗΜΟΚΡΙΤΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΡΑΚΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΑΘΗΜΑ
ΟΡΓΑΝΩΣΗ ΥΠΟΛΟΓΙΣΤΩΝ (206ΕΥΥΚ)
ΠΠΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΕΑΡΙΝΟ 2023-2024

Διάλεξη No4:
M68000: Σωρός - Stack

Δ. Καραμπατζάκης, Επίκουρος Καθηγητής

email. dkara@cs.ihu.gr

Δήλωση προσβασιμότητας

Σε αυτό το μάθημα όλες/οι οι φοιτήτριες/τές απολαμβάνουν – και αντίστοιχα υποχρεούνται να σέβονται – το δικαίωμα της ίσης μεταχείρισης. Δεν είναι ανεκτή και αποδεκτή κανενός τύπου και μορφής διάκριση με κριτήρια την εθνικότητα, τη φυλή, την καταγωγή, τη γλώσσα, το φύλο, τη θρησκεία, την ηλικία, την υγεία, τη σωματική ικανότητα, την ιδιωτική ζωή, τον γενετήσιο προσανατολισμό, τη σωματική ικανότητα και την οικονομική και κοινωνική κατάσταση στην οποία αυτοί βρίσκονται.

Το Πανεπιστήμιο άγρυπνα μεριμνά για τη διασφάλιση της αρχής των ίσων ευκαιριών και της ίσης μεταχείρισης. Οι κοινωνικές προκαταλήψεις και οι ιδεολογικές παρωπίδες είναι έννοιες τελείως ξένες με την επιστημονική πρόοδο την οποία το Πανεπιστήμιο είναι ταγμένο να υπηρετεί.

Ο Διδάσκων

Πληροφορίες για το Μάθημα

Διδάσκων:

Δημήτρης Καραμπατζάκης, Επίκουρος Καθηγητής

Αναλογικά και Ψηφιακά Ηλεκτρονικά Συστήματα

Μέλος Εργαστηρίου Βιομηχανικών και Εκπαιδευτικών Ενσωματωμένων Συστημάτων

Επικοινωνία / πληροφορίες:

Email. dkara@cs.ihu.gr

web. http://www.internetofthings.gr/

Ώρες Γραφείου:

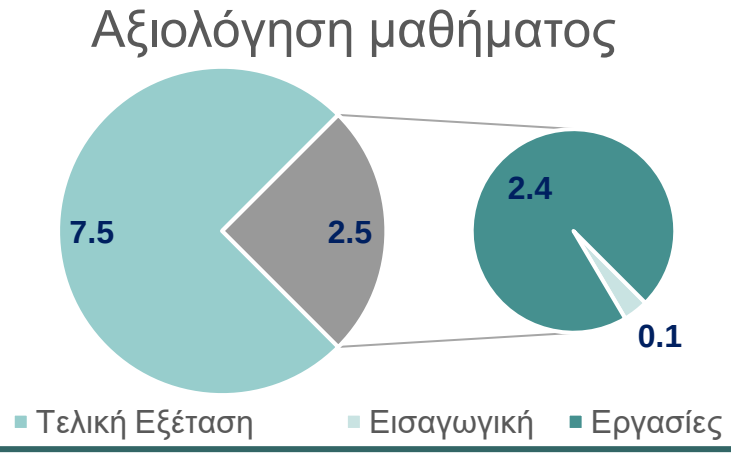
μετά από συνεννόηση με email στο ΦΕ 315 (πάνω από αιθ. Α1)

Πληροφορίες για το Μάθημα (Γενικές)

- **Κάθε Τρίτη, Πέμπτη 12.00 π.μ. - 14.00 μ.μ. μάθημα θεωρίας στο Μεγάλο Αμφιθέατρο (μπορεί να αλλάζει με ανακοινώσεις).**
- **Η διαχείριση του μαθήματος θα γίνει με χρήση της υπηρεσίας https://courses.cs.ihu.gr**
- **Όλοι οι φοιτητές πρέπει να έχουν λογαριασμό στο uregister.**
- **Η ιστοσελίδα με τις πληροφορίες του μαθήματος: http://iees.cs.ihu.gr/?page_id=3209**
- **Υλικό του μαθήματος στο moodle: https://moodle.cs.ihu.gr/**

Πληροφορίες για το Μάθημα (Αξιολόγηση)

- **Η βαθμολογία είναι 75% από την τελική εξέταση και 25% από τις ατομικές εργασίες (1 σετ ασκήσεων) που θα δοθούν για το σπίτι.**
- **Η τελική εξέταση είναι με ανοιχτό το κύριο σύγγραμμα του μαθήματος.**
- **Ο βαθμός του μαθήματος (ΒΜ = ΓΕ*0,75 + ΣΑ*0,25) πρέπει να είναι τουλάχιστον πέντε (5).**



Πληροφορίες για το Μάθημα (Μονάδες)

- **Κωδικός Μαθήματος:** 206ΕΥΥΚ
- **Εξάμηνο:** 2ο
- **Τύπος Μαθήματος:** Υποβάθρου, Ανάπτυξης Δεξιοτήτων
- **Είδος Μαθήματος:** Υποχρεωτικό (ΥΠ)
- **Διδασκαλία Θεωρίας:** 3 ώρες/εβδομάδα
- **Διδασκαλία Φροντιστήριο:** 1 ώρες/εβδομάδα
- **Πιστωτικές μονάδες ECTS: 7**
- **Γλώσσα διδασκαλίας και Εξετάσεων:** Ελληνικά

Πληροφορίες για το Μάθημα (Φόρτος)

Δραστηριότητα	Φόρτος εργασίας εξαμήνου
Διαλέξεις	78 ώρες
Φροντιστηριακές Ασκήσεις	26 ώρες
Γραπτές Εξετάσεις	2 ώρες
Γραπτές Εργασίες	34 ώρες
Αυτοτελής Μελέτη	35 ώρες
Σύνολο	**175 ώρες (7 ECTS)**

Κύριο Σύγγραμμα Μαθήματος (ΕΥΔΟΞΟΣ)

Οργάνωση και Σχεδίαση Υπολογιστών

Συγγραφέας: Πογαρίδης Δημήτριος

Έτος Έκδοσης: 2019

Κωδικός στον Εύδοξο: **86192986**

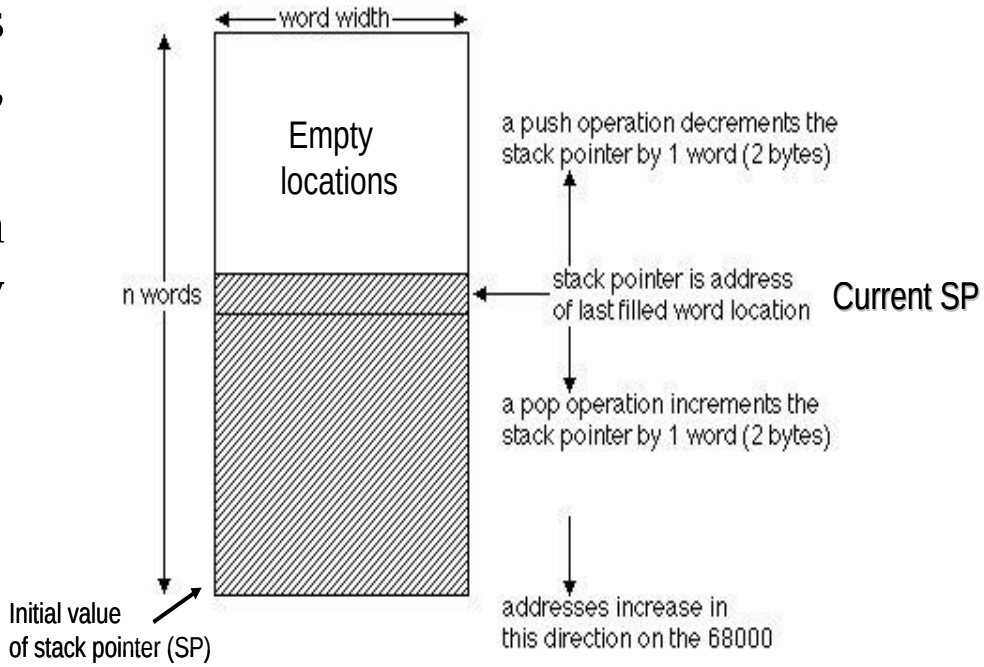
Λογισμικό - Αναπτυξιακό

- **Α’ μέρος μαθήματος (CISC):**
  - Assembly για τον Motorola68000
  - Λογισμικό easy68k http://www.easy68k.com/
- **Β’ μέρος μαθήματος (RISC):**
  - Υλοποίηση σχεδιάσεων σε αναπτυξιακό Arduino (προαιρετική αγορά του υλικού σύμφωνα με τις οδηγίες)
  - Λογισμικό Arduino IDE https://www.arduino.cc/en/Main/Software
  - Η γλώσσα προγραμματισμού (C++) και οι εντολές που υποστηρίζει είναι διαθέσιμες στο: https://www.arduino.cc/reference/en/

M68000
Σωρός – Στοίβα – Stack

Stacks

- **A stack is a Last In First Out (LIFO) buffer** containing a number of data items usually implemented as a block of **n consecutive bytes, words or long words in memory.**
- The address of the last data item placed into the stack **is pointed to by the Stack Pointer (SP).**
- **Application of stacks:**
  - Temporary storage of variables
  - Temporary storage of program addresses
  - Communication with subroutines



M68000 Stacks

- **In fact, stacks are nothing more than a region of memory. Any part of the linear memory space of the 68000 can be used as storage space for stacks.**
- **Stack addresses begin in high memory ($07FFE for example) and are pushed toward low memory ($07F00 for example). i.e. 68000 stacks grow into low memory.**
- **Other CPUs might do this in the reverse order (grow in high memory).**
- **Normally, address register A7 is used as a main stack pointer (SP) in the 68000. Using this register for other addressing purposes may lead to incorrect execution.**
- **68000 (Hardware) stack item size:**
  - **One word for data (by 2).**
  - **One longword for addresses (by 4).**
- **All you need to transform an ordinary section of memory into a stack is an address register to act as a stack pointer. User-defined stacks that use other item sizes (byte, word, longword), may be created by using address registers other than A7. Since the 68000 has 8 address registers, you can maintain up to 8 stacks simultaneously.**
- **A user defined stack pointer is incremented/decremented by values of 1, if the operations are byte operations, by 2 for words and by 4 for longwords.**

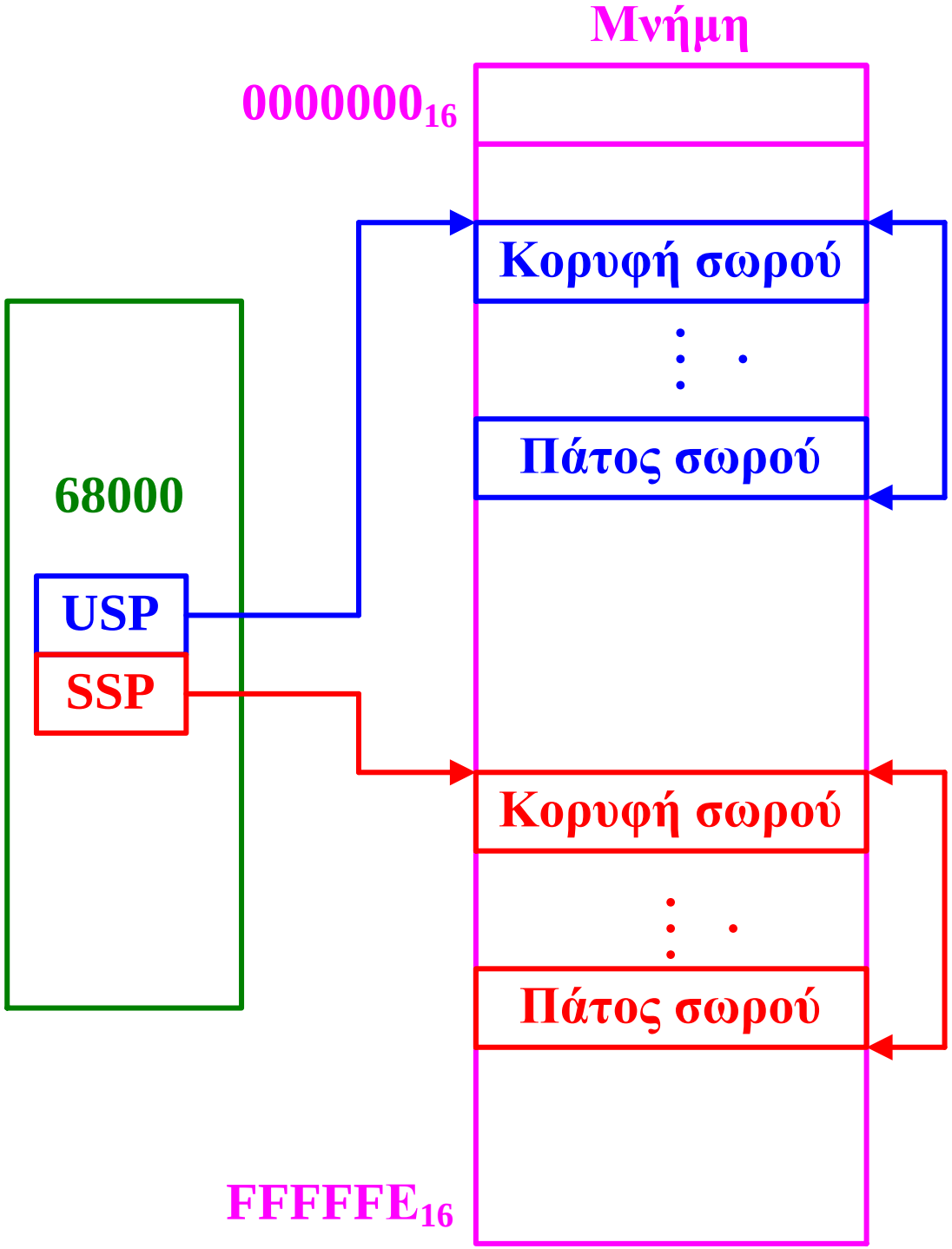
The Stack Pointer

- **A7 is a special address register, called the *stack pointer*.**
- **When programming assembly, we can use SP as an alias for A7.**

MOVEA.L #$3000,SP

- **It is also called USP (*User Stack* Pointer)**
- **There is also a supervisor stack pointer, but we won't worry about it yet.**
- **For each data element added to the Stack, the value of the SP is decremented by 2 or 4. Even if we place a byte of data on the Stack, the SP is always changed by 2, to allow a mixture of byte, word and longword data on the Stack.**
- **Data on the Stack should always be aligned at even byte boundaries.**

Δείκτες Σωρού Χρήστη και Επόπτη



- **Ο μικροεπεξεργαστής 68000 χρησιμοποιεί αρχιτεκτονική προσανατολισμένου σωρού.**

- **Ο 68000 διαθέτει δύο σωρούς και δύο δείκτες σωρού, *το δείκτη σωρού χρήστη (User Stack Pointer ή USP)* και *το δείκτη σωρού επόπτη (Supervisor Stack Pointer ή SSP).***

- **Και οι δύο σωροί μπορούν να φορτωθούν σε οποιοδήποτε μέρος της μνήμης και το μέγεθός τους δεν περιορίζεται.**

- **Ο δείκτης σωρού χρήστη είναι ενεργός όταν ο 68000 είναι σε κατάσταση χρήστη και ο δείκτης σωρού επόπτη είναι ενεργός όταν ο 68000 είναι σε κατάσταση επόπτη.**

- **Οι θέσεις μνήμης, γνωστές ως πάτος των σωρών, παριστάνουν τις θέσεις μνήμης που φορτώθηκαν αρχικά στους δείκτες σωρού. Όταν οι σωροί είναι άδειοι οι δείκτες δείχνουν αυτές τις θέσεις.**

Initializing the Stack Pointer

- **It's the programmer's responsibility to initialize the stack. This involves two (2) steps:**
  - **Initialize the stack pointer: The initial starting address or bottom of the stack.**
  - **Allocate sufficient memory for items to be pushed onto the stack. This could be done by locating the initial stack pointer at a very high memory address.**
- **Example:**

```
INITSP  EQU      $07FFE          Value of INITSP
        MOVEA.L  #INITSP,A7      Initialize  SP, A7
Or …
        MOVEA.L  #INITSP,SP      Initialize  SP
```

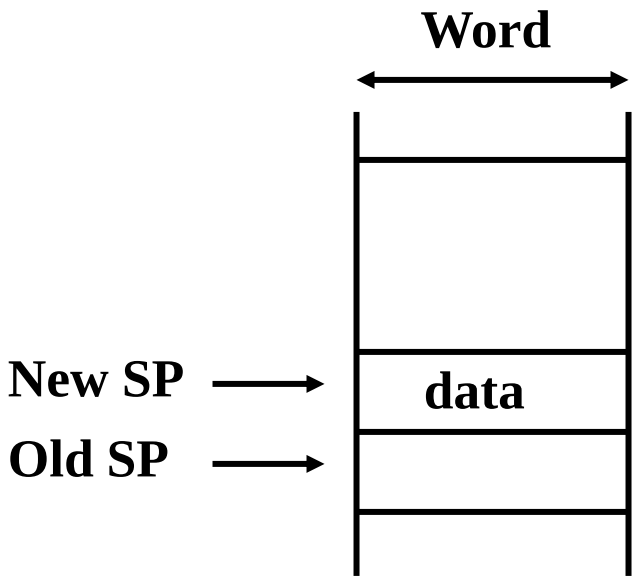
Push – Pop – Top

- **The stack grows upward toward the low address when items are pushed to the top of the stack.**
- **The stack pointer always points to the top item on the stack.**
- **When an item is pushed,**
  - **the stack pointer is decreased to point to the consecutive memory above**
  - **then the new item is added onto the stack**
- **When an item is popped,**
  - **the item on the top is copied to destination**
  - **then the stack pointer is increased to point to the consecutive memory below**

Stack Push Operations

- **To push an item onto the stack:**
  - **The stack pointer must be decremented by one word (i.e decremented by 2)**
- **We push values onto the stack using pre-decrement mode.**

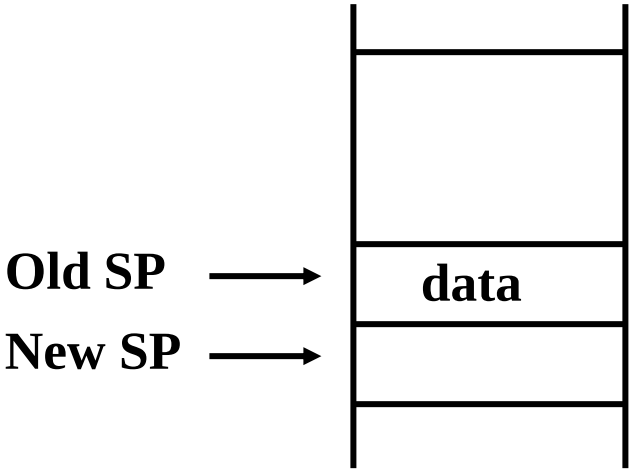
```
MOVE.W   D2,-(SP)
MOVE.W   D3,-(SP)
```



Stack Pop Operation

- **To pop an item off the stack:**
  - **The information or data is read from the stack.**
  - **The stack pointer incremented by one word**
- **We pop values from the stack using postincrement mode.**

```
MOVE.W   (SP)+, D3
MOVE.W   (SP)+, D2
```



Stack Top Operation

- **Top of the stack:**
  - **Top returns the value of a stack element without removing it from the stack.**
  - **Top(0) returns the top of the stack, Top(3) returns the 4th element in the stack.**
- **Top in 68k is implemented using ARI with Displacement.**

```
MOVE.L (12,SP),D3 or  MOVE.L 12(SP),D3
```

Other Instructions which affect the Stack

- **Special instruction MOVEM pushes multiple registers**

  MOVEM D0-D4/A0-A2,-(A7) for a push
  MOVEM (A7)+,D0-D4/A0-A2 for a pop

  - **Most used during procedure calls**
- **Another way to put things on the stack is with the PEA instruction.**

MOVE.W D2,-(SP)
MOVE.W D1,-(SP)
MOVE.W D0,-(SP)

- **Όλες οι παραπάνω εντολές σπρώχνουν δεδομένα μήκους λέξης στο σωρό.**
- **Στο σωρό μπορούν να σπρωχθούν και δεδομένα μήκους byte.**
- **Στην περίπτωση αυτή κάθε byte αποθηκεύεται στα οκτώ περισσότερο σημαντικά ψηφία της λέξης ενώ τα οκτώ λιγότερο σημαντικά ψηφία μένουν ανεπηρέαστα.**
- **Μπορούν επίσης να σπρωχθούν στο σωρό περιεχόμενα των εσωτερικών καταχωρητών του μικροεπεξεργαστή με την εντολή πολλαπλής μεταφοράς MOVEM.**

MOVEM.W D2/D1/D0, -(SP)

```
MOVE.W   (SP)+, D0
MOVE.W   (SP)+, D1
MOVE.W   (SP)+, D2
```

- **Όλες οι παραπάνω εντολές ανακαλούν δεδομένα μήκους λέξης απ' το σωρό με τον ίδιο τρόπο εισαγωγής τους στο σωρό.**

- **Μπορούν επίσης να ανακληθούν απ' το σωρό περιεχόμενα των εσωτερικών καταχωρητών του μικροεπεξεργαστή με την εντολή πολλαπλής μεταφοράς MOVEM.**

```
MOVEM.W (SP)+, D0/D1/D2
```

MOVEM

Syntax

MOVEM.s <ea>,<register list>

MOVEM.s <register list>,<ea>

Size = word, longword

<register list>:

1. **Rn – a single register**
2. **Rn-Rm – a range of registers (m>n)**
3. **Any combination of 1. and 2. separated by a slash /**

Examples

MOVEM.L D0-D7/A0-A6,$1234

MOVEM.L (A5),D0-D2/D5-D7/A0-A3/A6

MOVEM.W (A7)+,D0-D5/D7/A0-A6

MOVEM.W D0-D5/D7/A0-A6,-(A7)

A. a group of register is transferred to OR from Memory
(using AddrMode different than –(An) / (An)+)
MOVEM.L D0-D2/D4/A5/A6,$1234
Moves Regs D0,D1,D2,D4,A5,A6 to Memory, starting at location $1234 (D0) and moving to $1238
(.L +4 bytes, .W +2 bytes)
The order of transfer is D0 to D7, A0 to A7

B. If <ea> is –(An) only Reg to Mem operation is permitted
MOVEM.W D0-D5/D7/A0-A6,-(A7)
Moves Regs A6-A0,D7,D5-D0 to Memory,
starting at location (A7 minus .L 4 bytes, .W 2 bytes) and down through lower addresses (Multiple PUSH)
The order of transfer is A7 to A0, D7 to D0

C. If <ea> is (An)+ only Mem to Reg operation is permitted

MOVEM.W (A7)+,D0-D5/D7/A0-A6

Moves data from Mem to Regs D0-D5/D7/A0-A6, starting at Mem (A7 plus .L 4 bytes, .W 2 bytes) and up through higher addresses (Multiple POP)

The order of transfer is D0 to D7, A0 to A7

TIP: MOVEM.W sign-extends words when they are moved to Data Regs.

MOVEM: Instruction

Instruction: MOVEM.W D0-D3,(A0)

Register Contents

Before:

D0	55556666
D1	77778888
D2	9999AAAA
D3	BBBBCCCC
A0	000030B8

*****Memory*****

After :

Address	Content
0030B8	66 66
0030BA	88 88
0030BC	AA AA
0030BE	CC CC

Instruction: MOVEM.L D0-D3,(A0)

Register Contents

Before:

D0	55556666
D1	77778888
D2	9999AAAA
D3	BBBBCCCC
A0	000030B8

*****Memory*****

After :

Address	Content
0030B8	55 55
0030BA	66 66
0030BC	77 77
0030BE	88 88
0030C0	99 99
0030C2	AA AA
0030C4	BB BB
0030C6	CC CC

MOVEM: Instruction

Instruction: MOVEM.L (A0),D0-D3

Before: *****Memory*****

Address	Content
0030B8	11 11
0030BA	AA AA
0030BC	22 22
0030BE	BB BB
0030C0	33 33
0030C2	CC CC
0030C4	44 44
0030C6	DD DD

A0	000030B8

After : Register Contents

D0	1111AAAA
D1	2222BBBB
D2	3333CCCC
D3	4444DDDD

Instruction: MOVEM.W (A0),D0-D3

Before: *****Memory*****

Address	Content
0030B8	11 11
0030BA	AA AA
0030BC	22 22
0030BE	BB BB
0030C0	33 33
0030C2	CC CC
0030C4	44 44
0030C6	DD DD

A0	000030B8

After : Register Contents

D0	00001111
D1	FFFFAAAA
D2	00002222
D3	FFFFBBBB

Sign-extends

MOVEM.W $400500,D0/D5
MOVEM.W D0/D5,$400600

Πριν την εκτέλεση των εντολών

[$400500-1]=$3FFE, [$400502-3]=$F40B

[D0]=$HHHHHHHH, [D5]=$HHHHHHHH

Μετά την εκτέλεση της πρώτης εντολής

[D0]=$0000 3FFE, [D5]=$FFFF F40B

Μετά την εκτέλεση της δεύτερης εντολής

[$400600-1]=$3FFE, [$400602-3]=$F40B

PEA

- **Operation: [SP] <- [SP] – 4; [M([SP])] <- <ea>**
- **Syntax: PEA <ea>**
- **Size: longword**
- **It pushes an effective address on the stack, used when pushing pointers. This will decrease A7 with 4 (the size of a pointer).**
- **The difference between the LEA and PEA instructions is that LEA calculates an <ea> and puts it in an An, while PEA calcs an <ea> in the same way but pushes it on the Stack.**
- **PEA calcs an <ea> to be used later in ARI addressing. In particular, PEA facilitates the writing of position independent code.**

Παράδειγμα 3.6

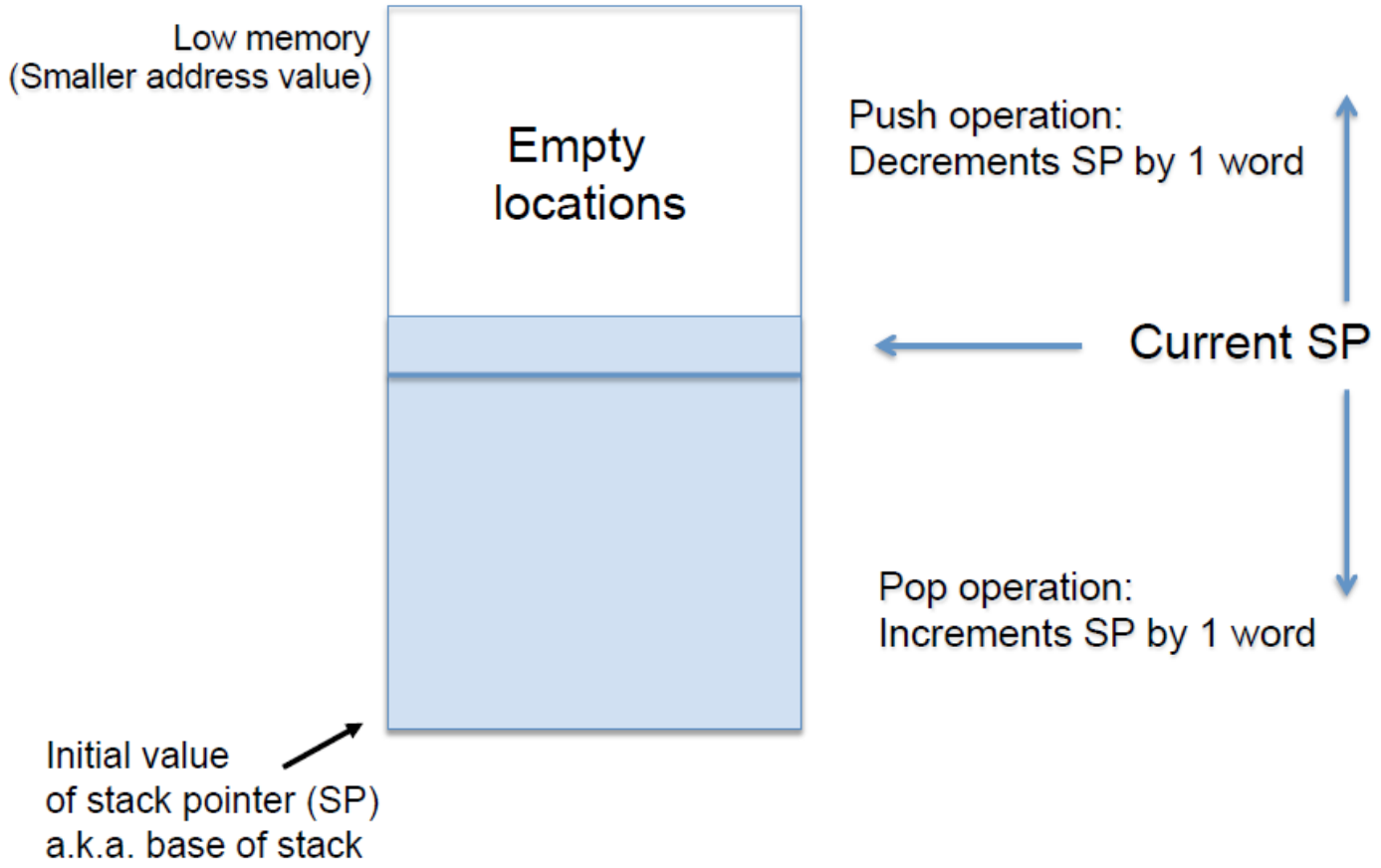
Να γραφτεί μια υπορουτίνα που:

1. Θα αποθηκεύει στο σωρό τα περιεχόμενα των καταχωρητών [D1]=$4035AD και [D2]=$4025BF.

2. Θα μεταφέρει τη μακριά λέξη της θέσης μνήμης [$400408]=$ADF56721 στη θέση μνήμης $40040C.

3. Θα ανακαλεί από το σωρό τα περιεχόμενα του καταχωρητή D1 και D2 και θα τα τοποθετεί στους καταχωρητές D4 και D3 αντίστοιχα.

a Stack


Low memory
(Smaller address value)
Empty
locations
Push operation:
Decrements SP by 1 word
Current SP
Pop operation:
Increments SP by 1 word
Initial value
of stack pointer (SP)
a.k.a. base of stack

```
                ORG $400400
NUM1            DC.L $4035AD
NUM2            DC.L $4025BF
NUM3            DC.L $ADF56721
NUM4            DC.L $37564D56

                ORG $400410
SUBRTN          MOVE .L NUM1,D1
                MOVE .L NUM2,D2
                MOVE.L  D1,-(SP)
                MOVE.L  D2,-(SP)
                MOVE.L NUM3,NUM4
                MOVE.L (SP)+,D3
                MOVE.L (SP)+,D4
                RTS
```

Μετά τη συμβολομετάφραση ο συμβολομεταφραστής θα δώσει την παρακάτω λίστα:

```
 1  00400400                          ORG     $400400
 2  00400400 004035AD      NUM1:      DC.L    $4035AD
 3  00400404 004025BF      NUM2:      DC.L    $4025BF
 4  00400408 ADF56721      NUM3:      DC.L    $ADF56721
 5  0040040C 37564D56      NUM4:      DC.L    $37564D56
 6
 7  00400410                          ORG     $400410
 8  00400410 223900400400  SUBRTN:    MOVE.L  NUM1,D1
 9  00400416 243900400404              MOVE.L  NUM2,D2
10  0040041C 2F01                      MOVE.L  D1,-(SP)
11  0040041E 2F02                      MOVE.L  D2,-(SP)
12  00400420 23F900400408              MOVE.L  NUM3,NUM4
         0040040C
13  0040042A 261F                      MOVE.L  (SP)+,D3
14  0040042C 281F                      MOVE.L  (SP)+,D4
15  00400410                          END     $400410
```

---------->MOVE.L $400400,D1

PC=400416 SR=2000 SS=00A00000 US=00000000 X=0

A0=00000000 A1=00000000 A2=00000000 A3=00000000 N=0

A4=00000000 A5=00000000 A6=00000000 A7=00A00000 Z=0

D0=00000000 D1=004035AD D2=00000000 D3=00000000 V=0

D4=00000000 D5=00000000 D6=00000000 D7=00000000 C=0

---------->MOVE.L $400404,D2

PC=40041C SR=2000 SS=00A00000 US=00000000 X=0

A0=00000000 A1=00000000 A2=00000000 A3=00000000 N=0

A4=00000000 A5=00000000 A6=00000000 A7=00A00000 Z=0

D0=00000000 D1=004035AD D2=004025BF D3=00000000 V=0

D4=00000000 D5=00000000 D6=00000000 D7=00000000 C=0

---------->MOVE.L D1,-(SP)

PC=40041E SR=2000 SS=009FFFFC US=00000000 X=0

09FFFFC:004035AD s

A0=00000000 A1=00000000 A2=00000000 A3=00000000 N=0

A4=00000000 A5=00000000 A6=00000000 A7=009FFFFC Z=0

D0=00000000 D1=004035AD D2=004025BF D3=00000000 V=0

D4=00000000 D5=00000000 D6=00000000 D7=00000000 C=0

---------->MOVE.L D2,-(SP)

PC=400420 SR=2000 SS=009FFFF8 US=00000000 X=0

009FFFF8:004025BF s 009FFFFC:004035AD s+4

A0=00000000 A1=00000000 A2=00000000 A3=00000000 N=0

A4=00000000 A5=00000000 A6=00000000 A7=009FFFF8 Z=0

D0=00000000 D1=004035AD D2=004025BF D3=00000000 V=0

D4=00000000 D5=00000000 D6=00000000 D7=00000000 C=0

---------->MOVE.L $400408,$40040C

PC=40042A SR=2008 SS=009FFFF8 US=00000000 X=0

009FFFF8:004025BF s 009FFFFC:004035AD s+4

A0=00000000 A1=00000000 A2=00000000 A3=00000000 N=1

A4=00000000 A5=00000000 A6=00000000 A7=009FFFF8 Z=0

D0=00000000 D1=004035AD D2=004025BF D3=00000000 V=0

D4=00000000 D5=00000000 D6=00000000 D7=00000000 C=0

α. Περιοχή δεδομένων μετά την εκτέλεση της εντολής:

400400 00 40 35 AD 00 40 25 BF AD F5 67 21 AD F5 67 21

---------->MOVE.L (SP)+,D3

PC=40042C SR=2000 SS=009FFFFC US=00000000 X=0
009FFFFC:004035AD s

A0=00000000 A1=00000000 A2=00000000 A3=00000000 N=0

A4=00000000 A5=00000000 A6=00000000 A7=009FFFFC Z=0

D0=00000000 D1=004035AD D2=004025BF D3=004025BF V=0

D4=00000000 D5=00000000 D6=00000000 D7=00000000 C=0

---------->MOVE.L (SP)+,D4

PC=40042E SR=2000 SS=00A00000 US=00000000 X=0

A0=00000000 A1=00000000 A2=00000000 A3=00000000 N=0

A4=00000000 A5=00000000 A6=00000000 A7=00A00000 Z=0

D0=00000000 D1=004035AD D2=004025BF D3=004025BF V=0

D4=004035AD D5=00000000 D6=00000000 D7=00000000 C=0