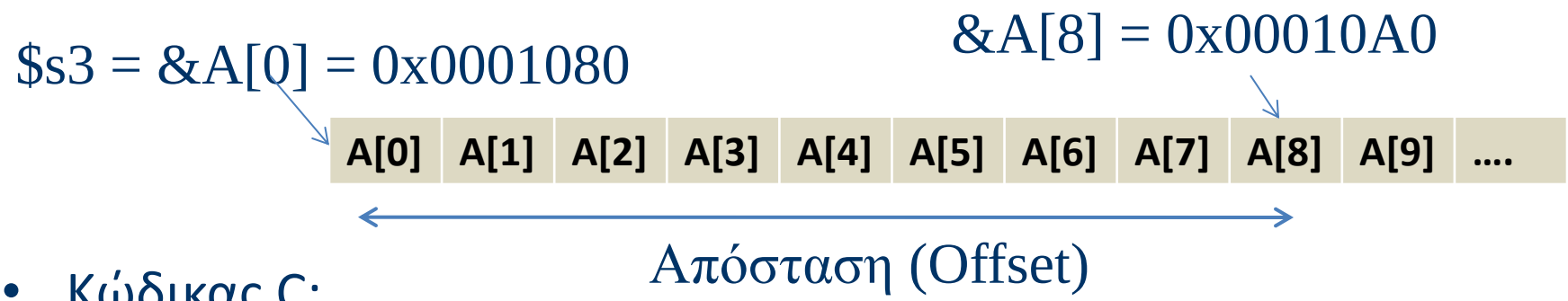
ΗΥ 134
Εισαγωγή στην Οργάνωση και στον Σχεδιασμό Υπολογιστών Ι

Διάλεξη 3
Εντολές του MIPS (2)

Νίκος Μπέλλας
Τμήμα Μηχανικών Η/Υ, Τηλεπικοινωνιών και Δικτύων

Παράδειγμα (συνέχεια από προηγ. διάλεξη)



- Κώδικας C:

```
g = h + A[8];
```

  - Το g είναι στον $s1, το h στον $s2, και η διεύθυνση (βάση) του A στον $s3
- Μεταγλωττισμένος κώδικας MIPS:
  - Η θέση 8 βρίσκεται 32 bytes μετά την αρχή του A
    - 4 bytes ανά λέξη

```
lw  $t0, 32($s3)     # load word
add $s1, $s2, $t0
```

Δεκαεξαεδικό (hex)

- Βάση 16
  - Περιεκτική αναπαράσταση
  - 4 bits για κάθε hex ψηφίο

0	0000	4	0100	8	1000	c	1100
1	0001	5	0101	9	1001	d	1101
2	0010	6	0110	a	1010	e	1110
3	0011	7	0111	b	1011	f	1111

- Παράδειγμα: eca8 6420
  - 1110 1100 1010 1000 0110 0100 0010 0000

Εντολές για Bytes/Halfwords

- MIPS byte/halfword load/store
  - Συνήθως για διαχείριση συμβολοσειρών (strings)

```
lb rt, offset(rs)    ; Επέκταση προσήμου  σε 32 bits στον rt
lh rt, offset(rs)

lbu rt, offset(rs)   ; Επέκταση 0 σε 32 bits στον rt
lhu rt, offset(rs)

sb rt, offset(rs)   ;Αποθηκεύει το δεξιότερο byte/halfword
sh rt, offset(rs)
```

Παραδείγματα εντολών μεταφοράς δεδομένων

Ας θεωρήσουμε τις παρακάτω θέσεις μνήμης και τα περιεχόμενα τους

ADDRESS	DATA
0x100080A2	0x21
0x100080A3	0x00
…	
0x10008104	0x01
0x10008105	0x11
0x10008106	0x33
0x10008107	0x80

Έστω ότι αρχικά:

$s0 = 0xAF109043
$s1 = 0x10008100

Επίσης ο MIPS είναι Big endian.

Εκτελείται η εντολή:
lw $s0, 4($s1)

Ποια ή νέα τιμή του $s0?

Εκτελείται η εντολή:
lw $s0, 5($s1)

ΛΑΘΟΣ (run-time error)
Πρόσβαση σε μη ευθυγραμμισμένη διεύθυνση (μη πολλαπλάσια του 4).

- Διεύθυνση Πρόσβασης : $s1 + $(4)_{10}$ = *0x10008104*
- Φορτώνουμε 4 bytes ξεκινώντας από την διεύθυνση *0x10008104*
- Τελικά ***$s0 = 0x01113380***

Παραδείγματα εντολών μεταφοράς δεδομένων

Ας θεωρήσουμε τις παρακάτω θέσεις μνήμης και τα περιεχόμενα τους

ADDRESS	DATA
0x100080A2	0x21
0x100080A3	0x00
…	
0x10008104	0x01
0x10008105	0x11
0x10008106	0x33
0x10008107	0x80

Έστω ότι αρχικά:

$s0 = 0xAF109043
$s1 = 0x10008100

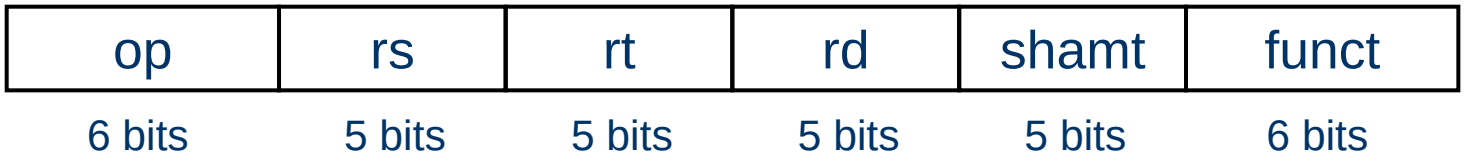
Επίσης ο MIPS είναι Big endian.

Εκτελείται η εντολή:
lb $s0, 7($s1)

Ποια ή νέα τιμή του $s0?

- Διεύθυνση Πρόσβασης : $s1 + $(7)_{10}$ = *0x10008107*
- Φορτώνουμε 1 byte από την διεύθυνση *0x10008107*. Πάντα στο Least Significant Byte του καταχωρητή.
- Κάνουμε επέκταση προσήμου για να γεμίσουμε (fill) τα 3 MSB του $s1
- Τελικά ***$s0 = 0xFFFFFF80***

MIPS Εντολές μορφής R



- Πεδία εντολής
  - op: opcode (κώδικας λειτουργίας)
  - rs: τελεστέος προέλευσης 1ου καταχωρητή
  - rt: τελεστέος προέλευσης 2ου καταχωρητή
  - rd: τελεστέος καταχωρητή προορισμού
  - shamt: ποσότητα ολίσθησης (shift amount) (00000 για τώρα)
  - funct: κωδικός συνάρτησης (function code)

Παράδειγμα μορφής R

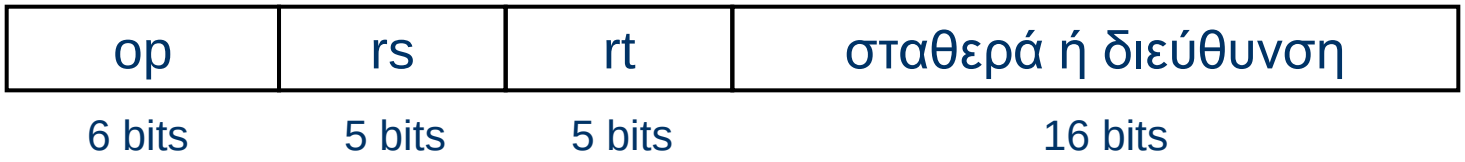
op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

```
add $t0, $s1, $s2
```

special	$s1	$s2	$t0	0	add
0	17	18	8	0	32
000000	10001	10010	01000	00000	100000

$00000010001100100100000000100000_2 = 02324020_{16}$

MIPS Εντολές μορφής I



- Άμεσες αριθμητικές εντολές ή εντολές μεταφοράς δεδομένων
  - rt: τελεστέος καταχωρητή προέλευσης ή προορισμού
  - Σταθερά: -2^{15} έως $+2^{15} - 1$
  - Διεύθυνση: προστίθεται απόσταση (offset) στη διεύθυνση βάσης που βρίσκεται στο rs

Πεδία του MIPS

- *Σχεδιαστική Αρχή :* Η καλή σχεδίαση απαιτεί καλούς συμβιβασμούς
  - Η ύπαρξη διαφορετικών μορφών περιπλέκει την αποκωδικοποίηση αλλα επιτρέπει ομοιομορφία στο μήκος των εντολών.

Λογικές πράξεις

- Εντολές για χειρισμό ακεραίων σε επίπεδο bit

Operation	C	Java	MIPS
Shift left	<<	<<	sll
Shift right	>>	>>>	srl
Bitwise AND	&	&	and, andi
Bitwise OR	|	|	or, ori
Bitwise NOT	~	~	nor

- Χρήσιμες για εξαγωγή ή εισαγωγή ομάδων από bits σε μια λέξη.

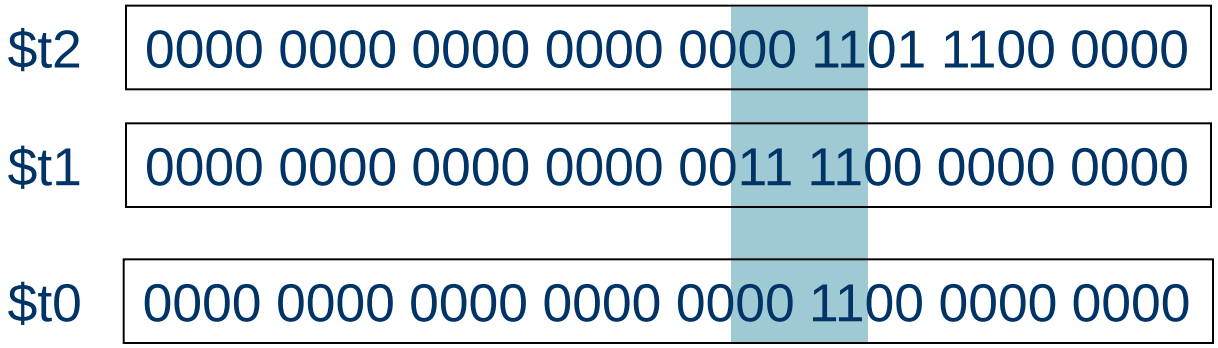
Πράξεις ολίσθησης (shift)

- shamt: ποσότητα (θέσεις) ολίσθησης
- Αριστερή ολίσθηση
  - Μεταφορά αριστερά και γέμισμα κενών με 0
  - `sll` κατά *i* bits πολλαπλασιάζει με 2^i
- Δεξιά ολίσθηση
  - Μεταφορά δεξιά και γέμισμα κενών με πρόσημο
  - `srl` κατά *i* bits διαιρεί με 2^i (μόνο απρόσημα)

Πράξη AND

- Χρήσιμη ως μάσκα ("καλύπτει" κάποια bits στη λέξη)
  - Επέλεξε κάποια bits, κάνε τα υπόλοιπα 0

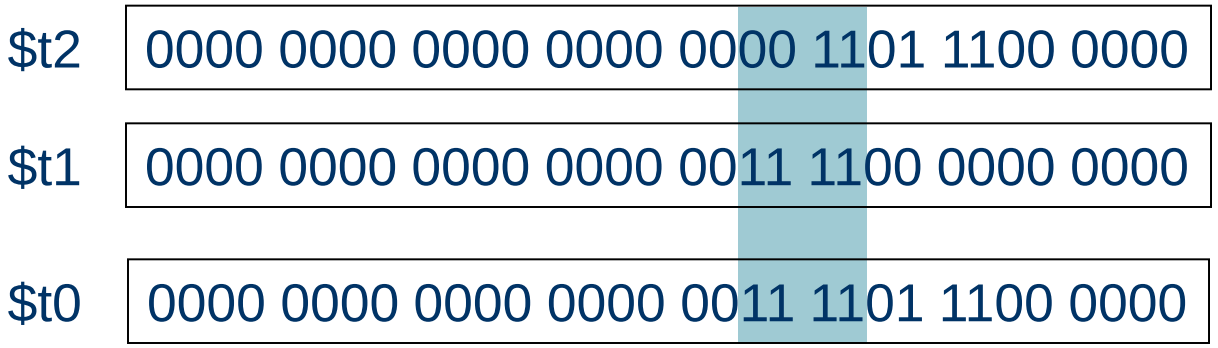
```
and $t0, $t1, $t2
```



Πράξη OR

- Χρήσιμη για να συμπεριλάβει κάποια bits σε μια λέξη
  - Κάνε κάποια bits 1, άσε τα υπόλοιπα ως έχουν

```
or $t0, $t1, $t2
```



Άμεσοι Τελεστέοι

- Σταθερά δεδομένα που εμφανίζονται σε μια εντολή

```
addi $s3, $s3, 4
```

- Δεν υπάρχει εντολή για άμεση αφαίρεση
  - Απλά χρησιμοποιείστε μια αρνητική σταθερά.

```
addi $s2, $s1, -1
```

- Ο καταχωρητής $0 του MIPS ($zero) είναι η σταθερά 0
  - Η τιμή του καταχωρητή αυτού δε μπορεί να διαγραφεί/μεταβληθεί
- Χρήσιμος για πολλές πράξεις
  - π.χ., μετακίνηση δεδομένων σε άλλο καταχωρητή

```
add $t2, $s1, $zero
```

Σχεδιαστική Αρχή

- Κάντε τη συνηθισμένη περίπτωση γρήγορη.
  - Μικρές σταθερές είναι συνήθεις
  - Με τη χρήση άμεσων τελεστέων αποφεύγουμε τη χρήση μιας επιπλέον εντολής load.

Σταθερές των 32-bit

- Οι περισσότερες σταθερές είναι μικρές
  - Άμεσοι τελεστέοι των 16-bit επαρκούν
- Για σταθερές των 32-bit:

  `lui rt, constant` (Load Upper Immediate)
  - Αντιγράφει τα ανώτερα 16 bits της σταθεράς constant στα αριστερά 16 bits του rt
  - Κάνει 0 τα άλλα 16 bits του rt και μετά μπορούμε να προσθέσουμε στο rt τα υπόλοιπα 16 bits του constant.

```
;store the 32-bit constant 0xAABBCCDD to $s0

addi $s0, $0, $0 ; $s0 = 0;
lui $s0, 0xAABB  ; $s0 = 0xAABB0000
ori $s0, 0xCCDD  ; $s0 = 0xAABBCCDD
```

Εντολές διακλάδωσης (I)

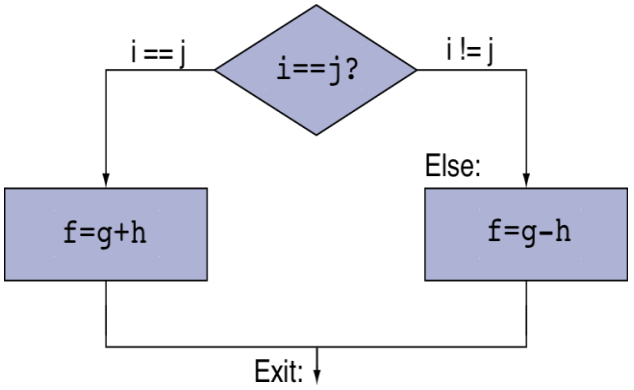
- Μετάβαση σε άλλη εντολή αν η συνθήκη είναι αληθής
  - Διαφορετικά, το πρόγραμμα συνεχίζει στην επόμενη εντολή
- `beq rs, rt, L1`
  - αν (rs == rt) πήγαινε στην εντολή με ετικέτα L1;
- `bne rs, rt, L1`
  - αν (rs != rt) πήγαινε στην εντολή με ετικέτα L1
- `j L1`
  - διακλάδωση χωρίς συνθήκη πήγαινε στην εντολή με ετικέτα L1

Μεταγλώττιση εντολών if

- κώδικας C :

```
if (i==j) f = g+h;
else f = g-h;
```

  - f, g,h… στους $s0, $s1, …



- Μεταγλωττισμένος κώδικας MIPS:

```
      bne $s3, $s4, Else
      add $s0, $s1, $s2
      j   Exit
Else: sub $s0, $s1, $s2
Exit: …
```

Ο assembler (συμβολομεταφραστής) υπολογίζει τις διευθύνσεις

Μεταγλώττιση βρόχων

- Κώδικας C:

```
int save[100];

while (save[i] == k) i++;
```

  - i στον $s3, k στον $s5, διεύθυνση του save στον $s6
- Μεταγλωττισμένος κώδικας MIPS:

```
Loop: sll  $t1, $s3, 2      # $t1=4*i
      add  $t1, $t1, $s6    # $t1=&save[i]
      lw   $t0, 0($t1)      # $t0=save[i]
      bne  $t0, $s5, Exit
      addi $s3, $s3, 1
      j    Loop
Exit: …
```

Εντολές διακλάδωσης (2)

- Θέσε το αποτέλεσμα σε 1 αν μια συνθήκη είναι αληθής, αλλιώς θέσε το σε 0
- `slt rd, rs, rt`
  - if (rs < rt) rd = 1; else rd = 0;
- `slti rt, rs, constant`
  - if (rs < constant) rt = 1; else rt = 0;
- Σε συνδυασμό με τις beq, bne

```
slt $t0, $s1, $s2  #  if ($s1 < $s2) $t0=1
bne $t0, $zero, L  #  branch to L
```

- Αντίστοιχη ψευδο-εντολή

```
blt $s1, $s2, L
```

Σχεδίαση εντολών διακλάδωσης

- Γιατί δεν υπάρχουν οι πραγματικές εντολές `blt`, `bge`, κτλ?
- Ο υπολογισμός του <, ≥, … είναι πιο αργός από τον υπολογισμό του =, ≠
  - Ο συνδυασμός σύγκρισης με διακλάδωση χρειάζεται περισσότερη δουλειά ανά εντολή και απαιτεί πιο αργό ρολόι.
  - Αυτό καθυστερεί όλες τις εντολές!
- `beq` και `bne` είναι οι πιο συνήθεις εντολές
- Αυτό είναι παράδειγμα συμβιβασμού (trade-off)

Ψευδοεντολές

- Οι περισσότερες εντολές του συμβολο-μεταφραστή έχουν μία προς μία αντιστοιχία με εντολές της γλώσσας μηχανής.
- Ψευδοεντολές: δεν είναι μέρος του συνόλου εντολών (ISA)

```
move $t0, $t1   →   add $t0, $zero, $t1
blt $t0, $t1, L →   slt $at, $t0, $t1
                    bne $at, $zero, L
```

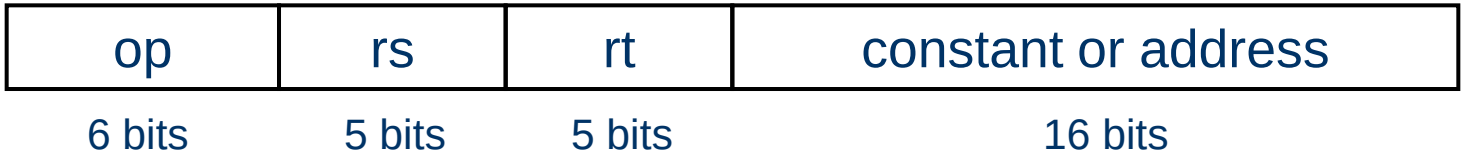
  - $at ($1): δεν μπορεί να χρησιμοποιηθεί από τον χρήστη, παρά μόνο από τον συμβολομεταφραστή

Προσημασμένο / Απρόσημο

- Προσημασμένη σύγκριση: `slt`, `slti`
- Απρόσημη σύγκριση: `sltu`, `sltui`
- Παράδειγμα:
  - $s0 = 1111 1111 1111 1111 1111 1111 1111 1111
  - $s1 = 0000 0000 0000 0000 0000 0000 0000 0001
  - `slt  $t0, $s0, $s1  # signed`
    - –1 < +1 ⇒ $t0 = 1
  - `sltu $t0, $s0, $s1  # unsigned`
    - +4,294,967,295 > +1 ⇒ $t0 = 0

Διευθυνσιοδότηση διακλαδώσεων

- Οι εντολές διακλαδώσεων προσδιορίζουν:
  - Opcode, δύο καταχωρητές, διεύθυνση προορισμού
  - Μορφή-I
- Ο προορισμός είναι συνήθως κοντά
  - Είτε πιο μπροστά είτε πιο πίσω
  - Η address δείχνει αριθμό 32-bit words , και όχι bytes



- Διευθυνσιοδότηση σχετική ως προς PC
  - Διεύθυνση προορισμού = (PC+4) + constant × 4
  - PC είναι η διεύθυνση της εντολής διακλάδωσης

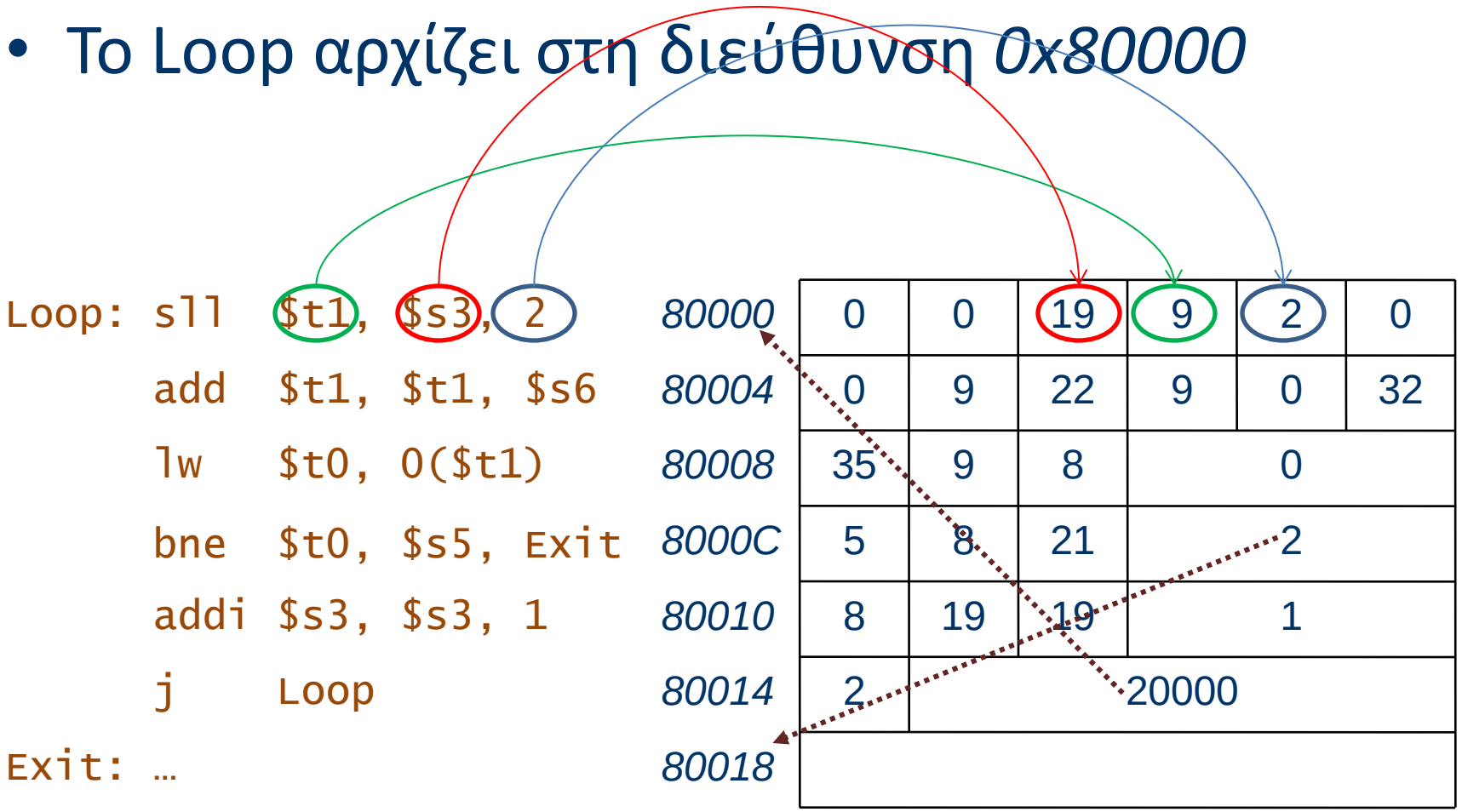
Διευθυνσιοδότηση αλμάτων

- Ο προορισμός ενός άλματος (`j` ή `jal`) μπορεί να είναι οπουδήποτε στο κείμενο
  - Χρειάζεται ολόκληρη η διεύθυνση στην εντολή
  - Μορφή-J

- (Ψευδο)απευθείας διευθυνσιοδότηση άλματος
  - Διεύθυνση προορισμού = $\{PC_{31...28}$, address × 4$\}$

Παράδειγμα

- Το Loop αρχίζει στη διεύθυνση *0x80000*



Μακρινές διακλαδώσεις

- Αν ο προορισμός μια διακλάδωσης είναι τόσο μακρυά ώστε 16 bit δεν είναι αρκετά για την απόσταση, τότε ο assembler μεταβάλει τον κώδικα
- Παράδειγμα

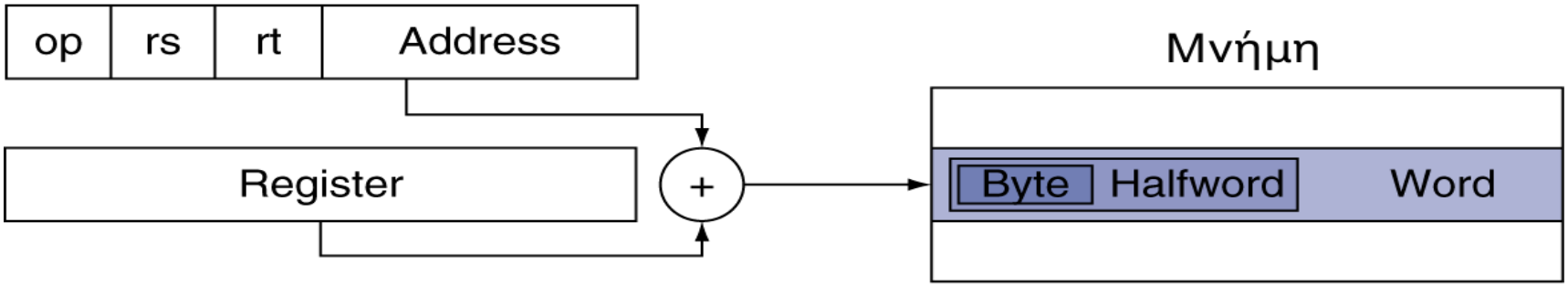
```
# L1 more than 2^15-1 words away
  beq $s0,$s1, L1
        ↓
   bne $s0,$s1, L2
   j L1
 L2: …
```

Περίληψη διευθυνσιοδότησης

1. Άμεση διευθυνσιοδότηση

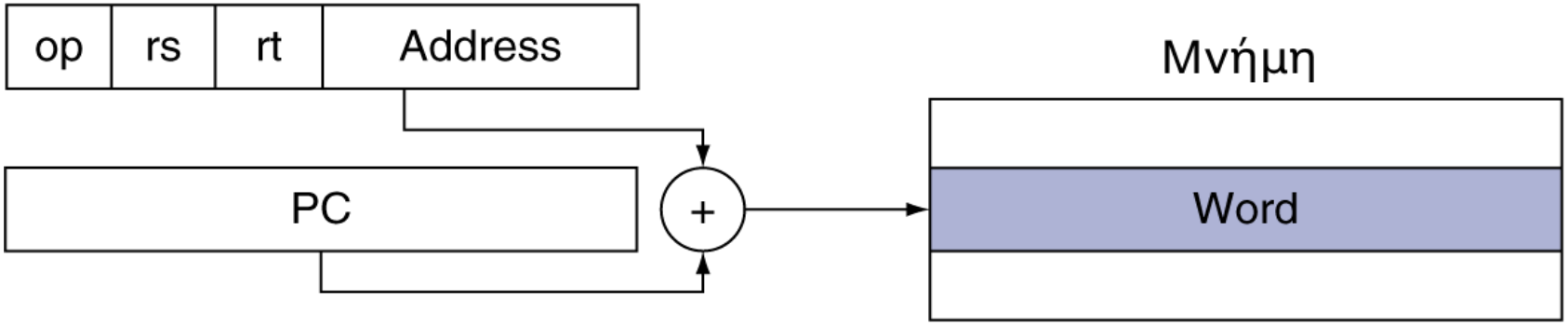
2. Διευθυνσιοδότηση μέσω καταχωρητή

3. Διευθυνσιοδότηση βάσης

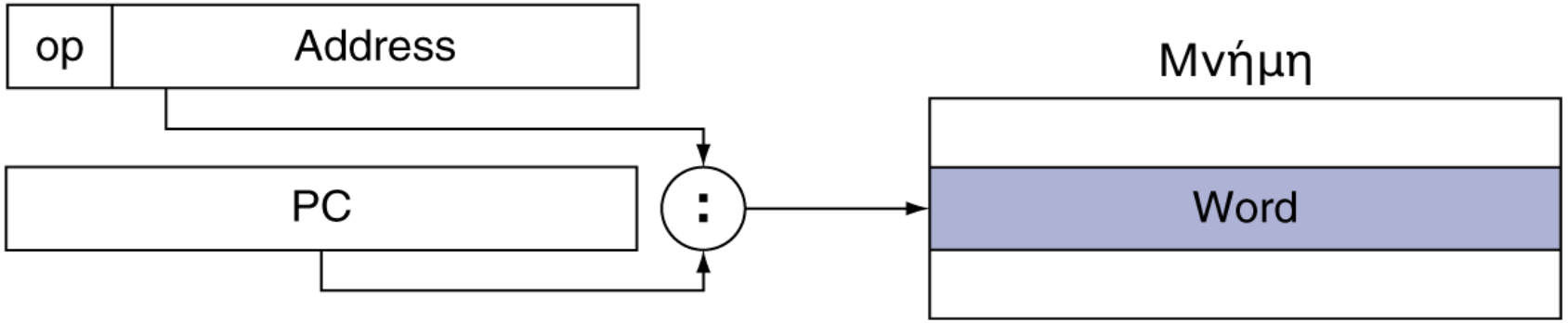


Περίληψη διευθυνσιοδότησης

4. Διευθυνσιοδότηση σχετική ως προς PC



5. Ψευδο-απευθείας διευθυνσιοδότηση



Διαδικασίες (procedures)

Γνωστές και σαν υπορουτίνες (subroutines)

- Τι πρέπει να προσέξουμε στις διαδικασίες (υπορουτίνες)?
  1. Η εμβέλεια των μεταβλητών που ορίζονται μέσα στις διαδικασίες (automatic variables) περιορίζεται στο σώμα των διαδικασιών αυτών
  2. Οι μεταβλητές αυτές (automatic variables), “κρύβουν” τον ορισμό των καθολικών μεταβλητών (global variables) με τις οποίες έχουν το ίδιο όνομα.