ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Προηγμένη Σχεδίαση Ψηφιακών Συστημάτων
Ψηφιακά Ολοκληρωμένα Κυκλώματα ΙΙ

Ενότητα 2

Αρχιτεκτονική Συνόλου Εντολών
Μελέτη του MIPS R2000

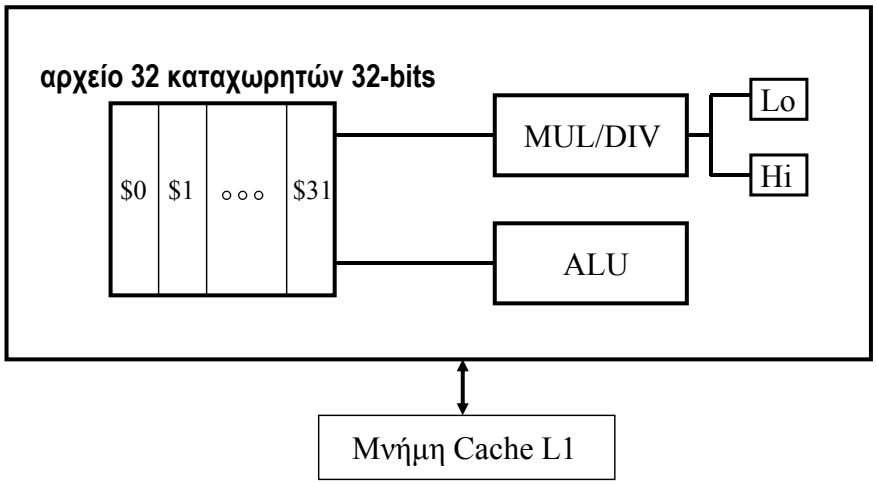
Καθηγητής Αντώνης Πασχάλης

2010

Γενικές Γραμμές

- **Τελεστέοι (operands) εντολών**
  - **Τύποι και Μεγέθη Τελεστέων**
  - **Οργάνωση των Bytes ενός Τελεστέου**
  - **Ευθυγράμμιση των Διευθύνσεων**
- **Χρήση της Μνήμης**
- **Αρχείο Καταχωρητών Γενικής Χρήσης**
- **Κωδικοποίηση Εντολών**
- **Τρόποι Διευθυνσιοδότησης**
- **Υποστηριζόμενες Εντολές του MIPS R2000**

MIPS R2000
Μονάδα Επεξεργασίας Ακεραίων - CPU

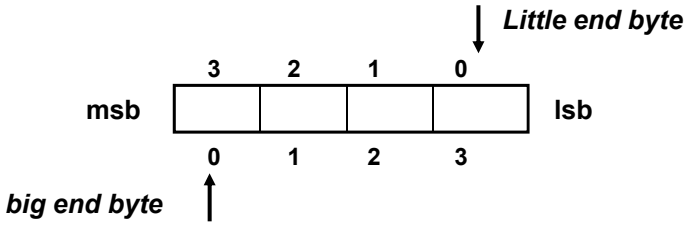


MIPS R2000: Τύποι & Μεγέθη Τελεστέων

- **Στο μάθημα αυτό ασχολούμαστε με την επεξεργασία ακεραίων αριθμών**
- **Οι αριθμητικές πράξεις (προσημασμένες και μη) εκτελούνται με τελεστέους (operands) που είναι προσημασμένοι ακέραιοι αριθμοί σε απεικόνιση συμπληρώματος ως προς 2 μεγέθους μίας λέξης (32 bits)**
- **Όταν οι ακέραιοι αριθμοί έχουν μέγεθος 8 ή 16 bits, τότε πριν την εκτέλεση της αριθμητικής πράξης γίνεται πάντα επέκταση πρόσημου, ανεξάρτητα από το εάν η πράξη είναι προσημασμένη ή όχι**
- **Όταν η αριθμητική πράξη είναι προσημασμένη (signed), γίνεται έλεγχος υπερχείλισης για προσημασμένους ακέραιους αριθμούς και προκαλείται η αντίστοιχη εξαίρεση (exception)**
  - **Συμβαίνει στις εντολές ADDI, ADD, SUB**
- **Όταν η αριθμητική πράξη δεν είναι προσημασμένη (unsigned), εκτελείται σαν προσημασμένη χωρίς να γίνεται έλεγχος υπερχείλισης**
  - **Συμβαίνει στις εντολές ADDIU, ADDU, SUBU**
  - **Χρησιμοποιούνται από τους μεταγλωττιστές της C**

Οργάνωση των Bytes ενός Τελεστέου

- **Big Endian: η διεύθυνση του τελεστέου είναι η διεύθυνση του περισσότερου σημαντικού byte του τελεστέου (xx00 = Big End Byte του τελεστέου)**
  - **IBM 360/370, Motorola 68000, MIPS, Sparc, HP PA**
- **Little Endian: η διεύθυνση του τελεστέου είναι η διεύθυνση του λιγότερου σημαντικού byte του τελεστέου (xx00 = Little End Byte του τελεστέου)**
  - **Intel 80x86, DEC Vax, DEC Alpha (Windows NT)**



Ευθυγράμμιση των Διευθύνσεων

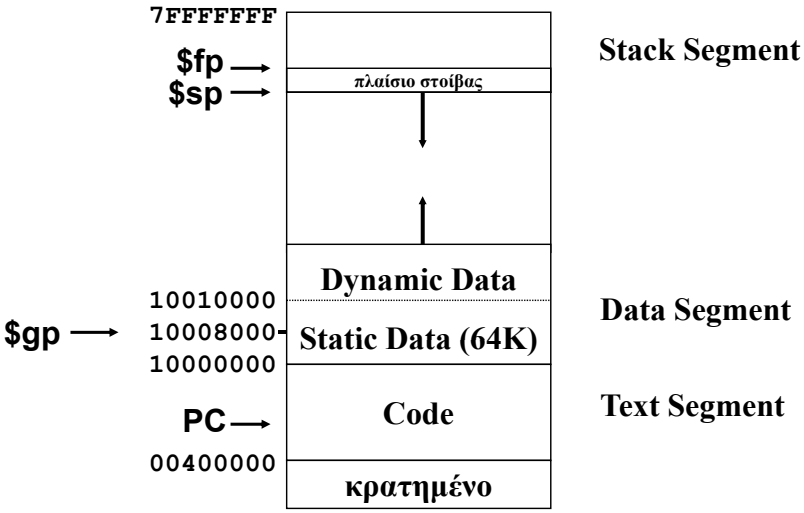
- **Ας υποθέσουμε ότι ένας τελεστέος έχει μέγεθος s Bytes**
- **Η διεύθυνση A της θέσης μνήμης στην οποία αποθηκεύεται ο τελεστέος είναι ευθυγραμμισμένη, εάν**

$$A \bmod s = 0$$

- **Η ευθυγράμμιση των διευθύνσεων προσφέρει:**
  - **πιο απλή υλοποίηση της διευθυνσιοδότησης στο υλικό**
  - **λιγότερες προσπελάσεις στη μνήμη**
  - **ταχύτερη εκτέλεση των προγραμμάτων**

Μεγέθη Τελεστέων (s)	Ευθυγράμμιση διεύθυνσης (στα 4 lsb της A = offset)
1 Byte (8 Bits)	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
2 Bytes (16 Bits)	0, 2, 4, 6, 8, A, C, E
4 Bytes (32 Bits)	0, 4, 8, C
8 Bytes (64 Bits)	0, 8

MIPS R2000: Χρήση της Μνήμης



MIPS R2000
Αρχείο Καταχωρητών Γενικής Χρήσης (ΚΓΧ)

όνομα	αριθμός	χρήση
$zero	0	Περιέχει τη σταθερή τιμή 0
$at	1	Δεσμευμένος από τον Assembler
$v0	2	Επιστροφή τιμών από functions
$v1	3	Επιστροφή τιμών από functions
$a0	4	Πέρασμα τιμών σε procedures*
$a1	5	Πέρασμα τιμών σε procedures
$a2	6	Πέρασμα τιμών σε procedures
$a3	7	Πέρασμα τιμών σε procedures

* περισσότερες τιμές περνούν μέσω στοίβας

MIPS R2000
Αρχείο Καταχωρητών Γενικής Χρήσης (ΚΓΧ)

όνομα	αριθμός	χρήση
$t0	8	Περιέχει δεδομένα που χανόνται*
$t1	9	Περιέχει δεδομένα που χάνονται
$t2	10	Περιέχει δεδομένα που χάνονται
$t3	11	Περιέχει δεδομένα που χάνονται
$t4	12	Περιέχει δεδομένα που χάνονται
$t5	13	Περιέχει δεδομένα που χάνονται
$t6	14	Περιέχει δεδομένα που χάνονται
$t7	15	Περιέχει δεδομένα που χάνονται

* δεν διατηρούνται μετά την κλήση διαδικασίας

MIPS R2000
Αρχείο Καταχωρητών Γενικής Χρήσης (ΚΓΧ)

όνομα	αριθμός	χρήση
$s0	16	Περιέχει δεδομένα που δεν χάνονται*
$s1	17	Περιέχει δεδομένα που δεν χάνονται*
$s2	18	Περιέχει δεδομένα που δεν χάνονται*
$s3	19	Περιέχει δεδομένα που δεν χάνονται*
$s4	20	Περιέχει δεδομένα που δεν χάνονται*
$s5	21	Περιέχει δεδομένα που δεν χάνονται*
$s6	22	Περιέχει δεδομένα που δεν χάνονται*
$s7	23	Περιέχει δεδομένα που δεν χάνονται*

*** διατηρούνται μετά την κλήση διαδικασίας**

MIPS R2000:
Αρχείο Καταχωρητών Γενικής Χρήσης (ΚΓΧ)

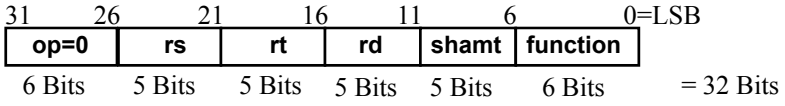
όνομα	αριθμός	χρήση
$t8	24	Περιέχει δεδομένα που χανόνται
$t9	25	Περιέχει δεδομένα που χάνονται
$k0	26	Δεσμευμένος από το Λειτουργικό
$k1	27	Δεσμευμένος από το Λειτουργικό
$gp	28	Καθολικός Δείκτης (Global Pointer)[1]
$sp	29	Δείκτης Στοίβας (Stack Pointer)[2]
$fp	30	Δείκτης Πλαισίου (Frame Pointer)[3]
$ra	31	Επιστροφή από διαδικασία

[1] δείχνει στο μέσο του static data segment για αποθήκευση καθολικών μεταβλητών
[2] δείχνει το τέλος της στοίβας με ευθυγράμμιση 2 λέξεων
[3] δείχνει την πρώτη λέξη του πλαισίου στοίβας της τρέχουσας διαδικασίας

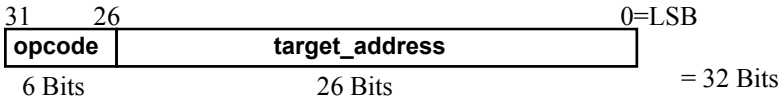
Κωδικοποίηση Εντολών MIPS R2000

- **Ο MIPS υποστηρίζει <u>σταθερή</u> κωδικοποίηση μεγέθους 32 bits με τρεις τύπους εντολών:**
  - **Τύπος I (immediate)**

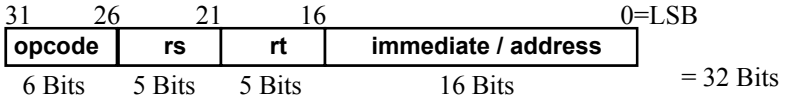
  - **Τύπος R (register)**



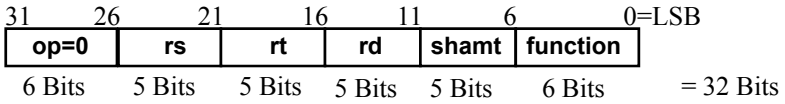
  - **Τύπος J (Jump)**



Κωδικοποίηση Εντολών MIPS R2000



Ίδιος κώδικας λειτουργίας (opcode = 0) για όλες τις εντολές τύπου R. Αξιοποιώντας τα 6 lsb της εντολής, που δεν χρησιμοποιούνται στις εντολές τύπου R επιτυγχάνεται μείωση του μεγέθους του opcode (μόνο 6 ψηφία) σε όφελος του πεδίου immediate



Κωδικοποίηση Εντολών MIPS R2000

- **Κωδικός λειτουργίας - opcode (bits 31-26)**
  (το πρώτο πεδίο της εντολής - Big Endian)
  - **χρησιμοποιείται για την αποκωδικοποίηση της εντολής**
  - **προσδιορίζει την πράξη - λειτουργία της εντολής (εκτός από τις εντολές τύπου Register)**
  - **προσδιορίζει τους τύπους και τα μεγέθη των τελεστέων**
  - **προσδιορίζει τον τρόπο διευθυνσιοδότησης μνήμης για την εύρεση της ενεργούς διεύθυνσης των τελεστέων**

Κωδικοποίηση Εντολών MIPS R2000

- **Πεδία διευθύνσεων καταχωρητών - registers**
  - **Προσδιορίζουν συγκεκριμένους καταχωρητές του αρχείου καταχωρητών ($0 - $31)**
  - **rs : register source (bits 25-21), προσδιορίζει τον πηγαίο καταχωρητή**
  - **rt : register target (bits 20-16), προσδιορίζει έναν καταχωρητή του οποίου η σημασία εξαρτάται από την εντολή (δεύτερος πηγαίος καταχωρητής, καταχωρητής προορισμού, ..)**
  - **rd : register destination (bits 15-11), προσδιορίζει τον καταχωρητή προορισμού, όπου αποθηκεύεται το αποτέλεσμα μίας πράξης**

Κωδικοποίηση Εντολών MIPS R2000

- **Πεδίο shamt (shift amount - bits 10-6)**
  - **Χρησιμοποιείται αποκλειστικά στις εντολές ολίσθησης με σταθερό αριθμό ολισθήσεων (SLL, SRL, SRA)**
  - **προσδιορίζει τον αριθμό των ολισθήσεων**
- **Πεδίο function (bits 5-0)**
  - **Χρησιμοποιείται αποκλειστικά στις εντολές τύπου Register ως πεδίο επέκτασης του κωδικού λειτουργίας**
  - **προσδιορίζει την πράξη - λειτουργία της εντολής**

Κωδικοποίηση Εντολών MIPS R2000

- **Πεδίο immediate (bits 15-0)**
  - **Χρησιμοποιείται αποκλειστικά στις εντολές τύπου Immediate**
  - **προσδιορίζει μία σταθερή τιμή στις αριθμητικές και λογικές πράξεις και εντολές σύγκρισης**
  - **προσδιορίζει τη μετατόπιση σε εντολές που χρησιμοποιούν διευθυνσιοδότηση με μετατόπιση και PC-σχετική διευθυνσιοδότηση**

Κωδικοποίηση Εντολών MIPS R2000

- **Πεδίο target_address (bits 25-0)**
  - **Χρησιμοποιείται αποκλειστικά στις εντολές τύπου Jump που χρησιμοποιούν τη ψευδο-άμεση (pseudo-direct) διευθυνσιοδότηση**
  - **προσδιορίζει τη διεύθυνση της επόμενης εντολής που θα εκτελεστεί λόγω της μεταπήδησης ως εξής:**
    - **τα 2 λιγότερο σημαντικά ψηφία της διεύθυνσης είναι 0 λόγω ευθυγράμμισης (εντολή 4 bytes)**
    - **τα 26 επόμενα λιγότερο σημαντικά ψηφία της διεύθυνσης προσδιορίζονται από το πεδίο target_address**
    - **τα 4 περισσότερο σημαντικά ψηφία της διεύθυνσης παραμένουν ως έχουν (όριο τα 256 MB = 64 M εντολές)**

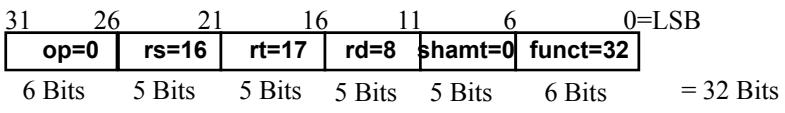
MIPS R2000: Τρόποι Διευθυνσιοδότησης Μνήμης Δεδομένων

- **Διευθυνσιοδότηση Μέσω Καταχωρητή (Register Addressing)**
- **Άμεση Διευθυνσιοδότηση (Immediate Addressing)**
- **Διευθυνσιοδότηση Μετατόπισης Displacement Addressing**
  - **Έμμεση Διευθυνσιοδότηση μέσω Καταχωρητή (Register Indirect Addressing)**
  - **Απόλυτη Διευθυνσιοδότηση Μνήμης (Absolute Addressing)**

Διευθυνσιοδότηση Μέσω Καταχωρητή Register Addressing MIPS R2000

- **Χρήση: όταν οι τελεστέοι βρίσκονται στους καταχωρητές**
  - **χρησιμοποιείται σε εντολές τύπου Register, που αφορούν κυρίως αριθμητικές και λογικές πράξεις**
  - **οι δύο τελεστέοι και το αποτέλεσμα της πράξης αποθηκεύονται σε τρεις καταχωρητές**
- **Πλεονεκτήματα: δεν απαιτείται προσπέλαση στη μνήμη - μικρό CPI(j)**
- **Μειονεκτήματα: περιορίζεται μόνο στους διαθέσιμους καταχωρητές γενικής χρήσης**

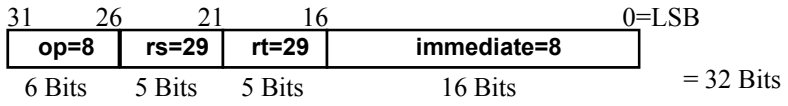
ADD $t0, $s0, $s1 $t0 <= $s0 + $s1 reg[08] <= reg[16] + reg[17]



Άμεση Διευθυνσιοδότηση Immediate Addressing MIPS R2000

- **Χρήση: όταν ο ένας τελεστέος είναι σταθερά το πολύ 16 bits**
  - **χρησιμοποιείται σε εντολές τύπου Immediate, που αφορούν αριθμητικές και λογικές πράξεις, αύξηση ενός δείκτη (arrays, loops, στοίβα), συγκρίσεις**
  - **η σταθερή τιμή αποθηκεύεται στο πεδίο immediate**
- **Πλεονεκτήματα: δεν απαιτείται προσπέλαση στη μνήμη**
- **Μειονεκτήματα: περιορισμός στο μέγεθος του σταθερού τελεστέου**

ADDI $sp, $sp, 8 $sp <= $sp + 8 reg[29] <= reg[29] + 8



Διευθυνσιοδότηση Μετατόπισης Displacement Addressing MIPS R2000

- **Χρήση: όταν γίνεται προσπέλαση της μνήμης**
  - **χρησιμοποιείται σε εντολές τύπου Immediate, που αφορούν φόρτωση από τη μνήμη (LOAD) και αποθήκευση στη μνήμη (STORE)**
  - **η ενεργή διεύθυνση προσδιορίζεται από το άθροισμα του περιεχομένου του καταχωρητή rs με τη σταθερή μετατόπιση το πολύ 16 bits που αποθηκεύεται στο πεδίο immediate**
  - **οι διευθύνσεις στη μνήμη έχουν μέγεθος 32 bits και είναι ευθυγραμμισμένες (προσπέλαση σε 2^{32} θέσεις μνήμης του ενός byte)**
  - **η οργάνωση των bytes των τελεστέων είναι Big Endian**
  - **υποστηρίζεται φόρτωση & αποθήκευση τελεστέων μεγέθους 1 byte (B), 2 bytes (H), 4 bytes (W) η οποία προσδιορίζεται στον κωδικό λειτουργίας opcode**

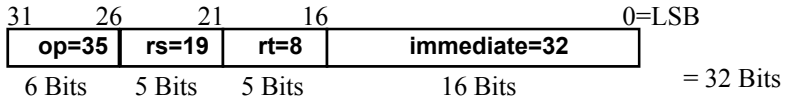
Διευθυνσιοδότηση Μετατόπισης Displacement Addressing MIPS R2000

- **Πλεονεκτήματα: ευελιξία**
- **Μειονεκτήματα: απαίτηση πρόσθεσης για τον υπολογισμό της ενεργούς διεύθυνσης**

LW $t0, 32($s3)

$t0 <= Mem[$s3 + 32]

reg[8] <= Mem[reg[19] + 32]



Άλλοι Τρόποι Διευθυνσιοδότησης MIPS R2000

- **Έμμεση διευθυνσιοδότηση μέσω καταχωρητή (reg. indirect)**
  - **προκύπτει από τη διευθυνσιοδότηση μετατόπισης, εάν θέσουμε σαν μετατόπιση το 0**
  - **Παράδειγμα: LW $t0,0($s2) = LW $t0,($s2)**
    (reg[8] <= Mem[reg[18]] (32 bits)
- **Απόλυτη διευθυνσιοδότηση μνήμης (absolute)**
  - **προκύπτει από τη διευθυνσιοδότηση μετατόπισης, εάν θέσουμε σαν μετατόπιση την απόλυτη διεύθυνση και σαν καταχωρητή που μετέχει στην διευθυνσιοδότηση το $zero**
  - **Παράδειγμα: LW $t0,1000($zero)**
    (reg[8] <= Mem[1000+0] (32 bits)

Προκύπτουν σαν ειδικές περιπτώσεις της διευθυνσιοδότησης μετατόπισης

Τυπικές Εντολές CPU MIPS R2000

Φόρτωσης και αποθήκευσης (Load & Store)	**Load (byte, half word, word) (από μνήμη σε καταχωρητή) Store (byte, half word, word) (από καταχωρητή στη μνήμη)**
Μεταφοράς Δεδομένων (Data Movement)	**Move (from hi, from lo, to hi, to lo) (από καταχωρητή σε καταχωρητή)**
Αριθμητικές και Λογικές (Arithmetic & Logic)	**integer Addition, Subtract, Multiply, Divide Shift left, Shift right arithmetic/logical And, Or, Nor, Xor**
Διαχείρισης σταθερών	**Load upper immediate**
Σύγκρισης	**Set less than**
Ελέγχου ροής προγράμματος	**Branch (eq, ne) Jump, Jump register (and link)**

Οι 10 πιο συχνές εντολές του 80x86

Σειρά	Εντολή	Ποσοστό % των συνολικά εκτελουμένων
1	load	22%
2	conditional branch	20%
3	compare	16%
4	store	12%
5	add	8%
6	and	6%
7	sub	5%
8	move register-register	4%
9	call	1%
10	return	1%
	Άθροισμα	96%

Οι απλές εντολές κυριαρχούν σε 5 προγράμματα SPECint92

Σημειογραφία

- **Σε κάθε διεύθυνση της μνήμης είναι αποθηκευμένο 1 byte.**
- **Το X σαν διεύθυνση μνήμης στην οποία είναι αποθηκευμένος ένας τελεστέος σημαίνει τη διεύθυνση μνήμης στην οποία είναι αποθηκευμένο το περισσότερο σημαντικό byte του τελεστέου (Big Endian)**
- **Στη διεύθυνση X+k-1 είναι αποθηκευμένο το λιγότερο σημαντικό byte του τελεστέου, όταν αυτός έχει μέγεθος k bytes**
- **Η τιμή του τελεστέου είναι Mem[X].**
- **Τα ψηφία ενός byte της μνήμης αριθμούνται έτσι, ώστε LSB=0 και MSB=7**
- **Reg[X] = το περιεχόμενο του καταχωρητή $X των 32 ψηφίων**
- **Τα ψηφία ενός καταχωρητή αριθμούνται έτσι, ώστε LSB=0 και MSB=31**

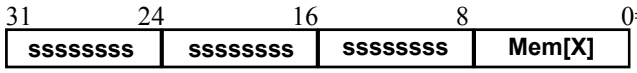
Εντολές Φόρτωσης & Αποθήκευσης (Τύπου Ι)

Εντολή	Περιγραφή
LB $s1,100($s2)	Load byte (με επέκταση πρόσημου)
LBU $s1,100($s2)	Load byte unsigned (με επέκταση μηδενός)
LH $s1,100($s2)	Load halfword (με επέκταση πρόσημου)
LHU $s1,100($s2)	Load halfword unsigned (με επέκταση μηδενός)
LW $s1,100($s2)	Load word
SB $s1,100($s2)	Store least significant byte (LSB)
SH $s1,100($s2)	Store least significant halfword (LSH)
SW $s1,100($s2)	Store word (W)

Εντολές Φόρτωσης & Αποθήκευσης Ακεραίων (Τύπου I)

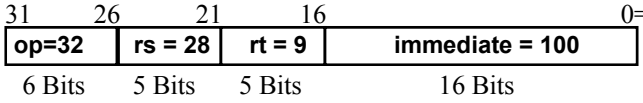
- `LB rt,immediate(rs) op=0x20`
  - **address = sign_ext(immediate) + Reg[rs]**
    Reg[rt] = sign_ext(Mem[address])
  - **Παράδειγμα :** `LB $t1,100($gp)`

Φόρτωση στον $9 του τελεστέου μεγέθους 8 ψηφίων μετά από επέκταση πρόσημου, που βρίσκεται σε 1 byte της μνήμης στη διεύθυνση X = 100+Reg[28]



($9 = $t1)

Το περιεχόμενο του $9 μετά την εκτέλεση της εντολής, όπου s το πρόσημο (bit 7)



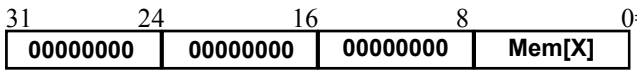
(IR) = 32 Bits

(κωδικοποίηση της εντολής του παραδείγματος)

Εντολές Φόρτωσης & Αποθήκευσης Ακεραίων (Τύπου I)

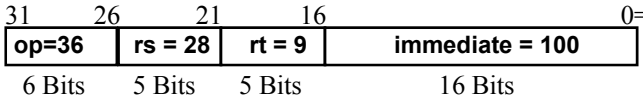
- `LBU rt,immediate(rs) op=0x24`
  - **address = sign_ext(immediate) + Reg[rs]**
    Reg[rt] = zero_ext(Mem[address])
  - **Παράδειγμα :** `LBU $t1,100($gp)`

Φόρτωση στον $9 του τελεστέου μεγέθους 8 ψηφίων μετά από επέκταση μηδενός, που βρίσκεται σε 1 byte της μνήμης στη διεύθυνση X = 100+Reg[28]



($9 = $t1)

Το περιεχόμενο του $9 μετά την εκτέλεση της εντολής



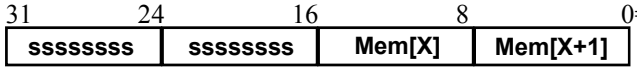
(IR) = 32 Bits

(κωδικοποίηση της εντολής του παραδείγματος)

Εντολές Φόρτωσης & Αποθήκευσης Ακεραίων (Τύπου I)

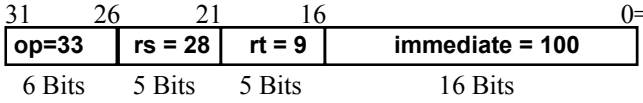
- **`LH rt,immediate(rs) op=0x21`**
  - **address = sign_ext(immediate) + Reg[rs]**
    Reg[rt] = sign_ext(Mem[address])
  - **Παράδειγμα : `LH $t1,100($gp)`**

Φόρτωση στον $9 του τελεστέου μεγέθους 16 ψηφίων μετά από επέκταση πρόσημου, που βρίσκεται σε 2 διαδοχικά bytes της μνήμης στη διεύθυνση X = 100+Reg[28]



($9 = $t1)

Το περιεχόμενο του $9 μετά την εκτέλεση της εντολής, όπου s το πρόσημο (bit 15)



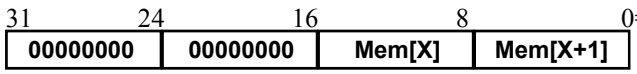
(IR) = 32 Bits

(κωδικοποίηση της εντολής του παραδείγματος)

Εντολές Φόρτωσης & Αποθήκευσης Ακεραίων (Τύπου I)

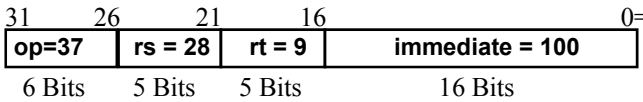
- **`LHU rt,immediate(rs) op=0x25`**
  - **address = sign_ext(immediate) + Reg[rs]**
    Reg[rt] = zero_ext(Mem[address])
  - **Παράδειγμα : `LBU $t1,100($gp)`**

Φόρτωση στον $9 του τελεστέου μεγέθους 16 ψηφίων μετά από επέκταση μηδενός, που βρίσκεται σε 2 διαδοχικά bytes της μνήμης στη διεύθυνση X = 100+Reg[28]



($9 = $t1)

Το περιεχόμενο του $9 μετά την εκτέλεση της εντολής



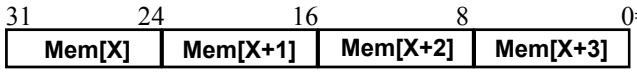
(IR) = 32 Bits

(κωδικοποίηση της εντολής του παραδείγματος)

Εντολές Φόρτωσης & Αποθήκευσης Ακεραίων (Τύπου I)

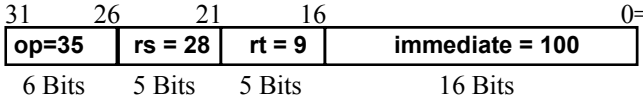
- `LW rt,immediate(rs) op=0x23`
  - address = sign_ext(immediate) + Reg[rs]
    Reg[rt] = Mem[address]
  - Παράδειγμα : `LW $t1,100($gp)`

Φόρτωση στον $9 του τελεστέου μεγέθους 32 ψηφίων,
που βρίσκεται σε 4 διαδοχικά bytes της μνήμης στη διεύθυνση X = 100+Reg[28]



($9 = $t1)

Το περιεχόμενο του $9 μετά την εκτέλεση της εντολής



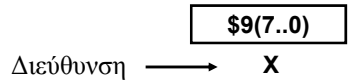
(IR) = 32 Bits

(κωδικοποίηση της εντολής του παραδείγματος)

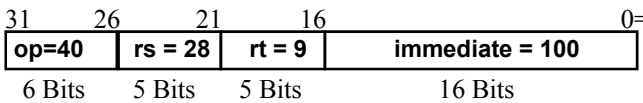
Εντολές Φόρτωσης & Αποθήκευσης Ακεραίων (Τύπου I)

- `SB rt,immediate(rs) op=0x28`
  - address = sign_ext(immediate) + Reg[rs]
    Mem[address] = Reg[rt] (low byte)
  - Παράδειγμα : `SB $t1,100($gp)`

Αποθήκευση στη μνήμη στη διεύθυνση X = 100+Reg[28]
του τελεστέου μεγέθους 8 ψηφίων, που βρίσκεται στα 8 LSB του $t1 = $9



Το περιεχόμενο της θέσης μνήμης μετά την εκτέλεση της εντολής



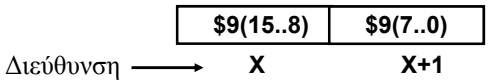
(IR) = 32 Bits

(κωδικοποίηση της εντολής του παραδείγματος)

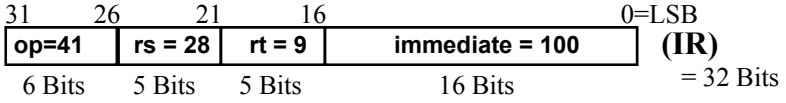
Εντολές Φόρτωσης & Αποθήκευσης Ακεραίων (Τύπου I)

- **`SH rt,immediate(rs) op=0x29`**
  - **address = sign_ext(immediate) + Reg[rs]**
    Mem[address] = Reg[rt] (low halfword)
  - **Παράδειγμα : `SH $t1,100($gp)`**

Αποθήκευση στη μνήμη στη διεύθυνση X = 100+Reg[28]
του τελεστέου μεγέθους 16 ψηφίων, που βρίσκεται στα 16 LSB του $t1 = $9



Το περιεχόμενο των 2 διαδοχικών θέσεων μνήμης μετά την εκτέλεση της εντολής

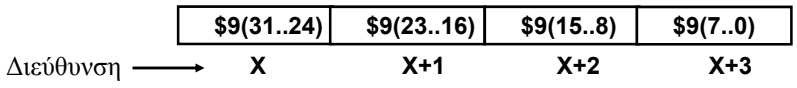


(κωδικοποίηση της εντολής του παραδείγματος)

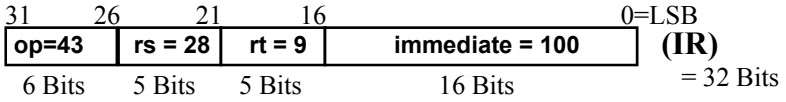
Εντολές Φόρτωσης & Αποθήκευσης Ακεραίων (Τύπου I)

- **`SW rt,immediate(rs) op=0x2B`**
  - **address = sign_ext(immediate) + Reg[rs]**
    Mem[address] = Reg[rt] (word)
  - **Παράδειγμα : `SW $t1,100($gp)`**

Αποθήκευση στη μνήμη στη διεύθυνση X = 100+Reg[28]
του τελεστέου μεγέθους 32 ψηφίων, που βρίσκεται στον $t1 = $9



Το περιεχόμενο των 4 διαδοχικών θέσεων μνήμης μετά την εκτέλεση της εντολής



(κωδικοποίηση της εντολής του παραδείγματος)

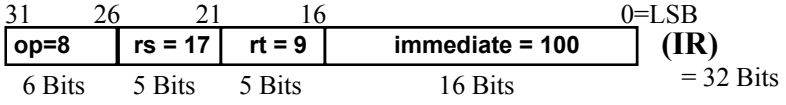
Αριθμητικές και Λογικές Εντολές (Τύπου Ι)

Εντολή	Περιγραφή
ADDI $t1,$s1,3	Addition immediate (με επέκταση πρόσημου – με υπερχείλιση)
ADDIU $t1,$s1,3	Addition immediate (με επέκταση πρόσημου – χωρίς υπερχείλιση)
ANDI $t1,$s1,3	AND immediate (με επέκταση μηδενός)
ORI $t1,$s1,3	OR immediate (με επέκταση μηδενός)
XORI $t1,$s1,3	XOR immediate (με επέκταση μηδενός)

Αριθμητικές και Λογικές Εντολές (Τύπου I)

- `ADDI rt, rs, immediate op=0x08`
  - **Reg[rt] = Reg[rs] + sign_ext(immediate)**
  - **Παράδειγμα :** `ADDI $t1, $s1, 100`

Αποθήκευση στον καταχωρητή $t1 = $9 του αθροίσματος του καταχωρητή $s1 = $17 με το περιεχόμενο του πεδίου immediate, αφού έχει υποστεί επέκταση πρόσημου. Λαμβάνεται υπόψη η υπερχείλιση.

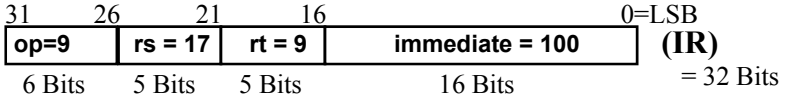


(κωδικοποίηση της εντολής του παραδείγματος)

Αριθμητικές και Λογικές Εντολές (Τύπου I)

- `ADDIU rt, rs, immediate op=0x09`
  - **Reg[rt] = Reg[rs] + sign_ext(immediate)**
  - **Παράδειγμα :** `ADDIU $t1, $s1, 100`

Αποθήκευση στον καταχωρητή $t1 = $9 του αθροίσματος του καταχωρητή $s1 = $17 με το περιεχόμενο του πεδίου immediate, αφού έχει υποστεί επέκταση πρόσημου. <u>Δεν</u> λαμβάνεται υπόψη η υπερχείλιση.

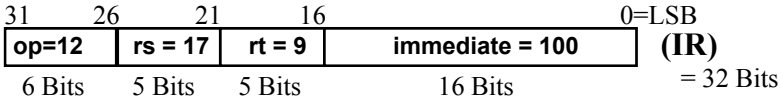


(κωδικοποίηση της εντολής του παραδείγματος)

Αριθμητικές και Λογικές Εντολές (Τύπου I)

- `ANDI rt, rs, immediate op=0x0C`
  - **Reg[rt] = Reg[rs] AND zero_ext(immediate)**
  - **Παράδειγμα :** `ANDI $t1, $s1, 100`

Αποθήκευση στον καταχωρητή $t1 = $9 της λογικής πράξης AND ανά ψηφίο του καταχωρητή $s1 = $17 με το περιεχόμενο του πεδίου immediate, αφού έχει υποστεί επέκταση μηδενός.

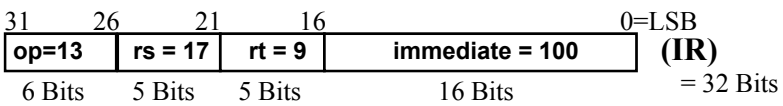


(κωδικοποίηση της εντολής του παραδείγματος)

Αριθμητικές και Λογικές Εντολές (Τύπου I)

- `ORI rt, rs, immediate op=0x0D`
  - **Reg[rt] = Reg[rs] OR zero_ext(immediate)**
  - **Παράδειγμα :** `ORI $t1, $s1, 100`

Αποθήκευση στον καταχωρητή $t1 = $9 της λογικής πράξης OR ανά ψηφίο του καταχωρητή $s1 = $17 με το περιεχόμενο του πεδίου immediate, αφού έχει υποστεί επέκταση μηδενός.

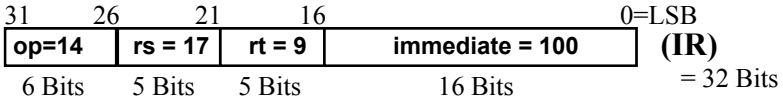


(κωδικοποίηση της εντολής του παραδείγματος)

Αριθμητικές και Λογικές Εντολές (Τύπου I)

- `XORI rt, rs, immediate op=0x0E`
  - **Reg[rt] = Reg[rs] XOR zero_ext(immediate)**
  - **Παράδειγμα :** `XORI $t1, $s1, 100`

Αποθήκευση στον καταχωρητή $t1 = $9 της λογικής πράξης XOR ανά ψηφίο του καταχωρητή $s1 = $17 με το περιεχόμενο του πεδίου immediate, αφού έχει υποστεί επέκταση μηδενός.



(κωδικοποίηση της εντολής του παραδείγματος)

Εντολές Διαχείρισης Σταθερών (Τύπου Ι)

Εντολή	Περιγραφή
LUI $t1,100	Load upper immediate

Παρέχει δυνατότητα αποθήκευσης μίας σταθεράς 32 ψηφίων σε καταχωρητή του αρχείου καταχωρητών, έστω $s0, σε 2 βήματα:

***Βήμα 1*: Άμεση φόρτωση σταθεράς 16 ψηφίων στο πάνω μισό του $t0**

```
LUI $t0, high_half
```

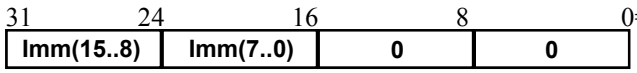
***Βήμα 2*: Φόρτωση σταθεράς 32 ψηφίων στον $s0**

```
XORI $s0,$t0,low_half
```

Εντολές Διαχείρισης Σταθερών (Τύπου Ι)

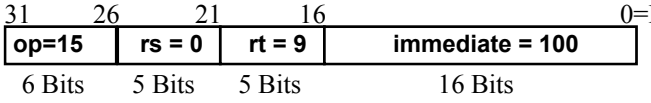
- `LUI rt, immediate op=0x0F`
  - **Reg[rt] = Reg[zero] XOR zero_ext(immediate)**
  - **Reg[rt] = Reg[rt] shifted left logical by 16**
  - **Παράδειγμα :** `LUI $t1, 100`

Αποθήκευση στ α 16 MSB του καταχωρητή $t1 = $9 του περιεχομένου του πεδίου immediate. Τα 16 LSB του καταχωρητή $t1 = $9 είναι 0.



($9 = $t1)

Το περιεχόμενο του $9 μετά την εκτέλεση της εντολής



(IR)
= 32 Bits

(κωδικοποίηση της εντολής του παραδείγματος)

Μεταφοράς Δεδομένων (Τύπου R)

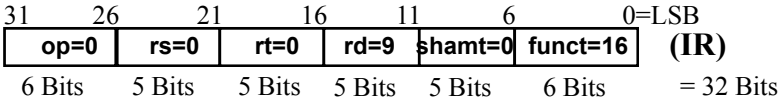
Εντολή	Περιγραφή
MFHI $t1	Move from Hi
MFLO $t1	Move from Lo
MTHI $t1	Move to Hi
MTLO $t1	Move to Lo

Η μονάδα πολλαπλασιασμού και διαίρεσης παράγει το διπλάσιο σε μέγεθος γινόμενο καθώς, και το πηλίκο και το υπόλοιπο στους επιπλέον καταχωρητές Lo και Hi, αντίστοιχα

Μεταφοράς Δεδομένων (Τύπου R)

- **`MFHI rd op=0x00 funct=0x10`**
  - **Reg[rd] = Hi**
  - **Παράδειγμα : `MFHI $t1`**

Μεταφορά στον καταχωρητή $t1 = $9 του περιεχομένου του καταχωρητή Hi.

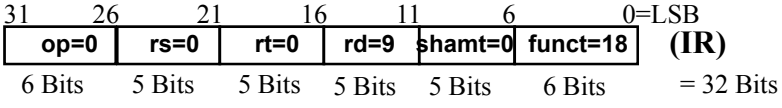


(κωδικοποίηση της εντολής του παραδείγματος)

Μεταφοράς Δεδομένων (Τύπου R)

- **`MFLO rd op=0x00 funct=0x12`**
  - **Reg[rd] = Lo**
  - **Παράδειγμα : `MFLO $t1`**

Μεταφορά στον καταχωρητή $t1 = $9 του περιεχομένου του καταχωρητή Lo.

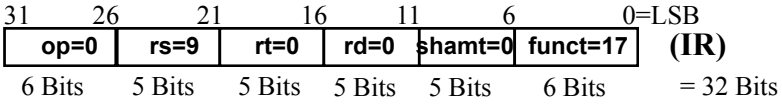


(κωδικοποίηση της εντολής του παραδείγματος)

Μεταφοράς Δεδομένων (Τύπου R)

- **`MTHI rs op=0x00 funct=0x11`**
  - **Hi = Reg[rs]**
  - **Παράδειγμα : `MTHI $t1`**

Μεταφορά στον καταχωρητή Hi του περιεχομένου του καταχωρητή $t1 = $9.

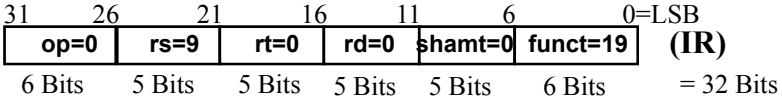


(κωδικοποίηση της εντολής του παραδείγματος)

Μεταφοράς Δεδομένων (Τύπου R)

- **`MTLO rs op=0x00 funct=0x13`**
  - **Lo = Reg[rs]**
  - **Παράδειγμα : `MTLO $t1`**

Μεταφορά στον καταχωρητή Lo του περιεχομένου του καταχωρητή $t1 = $9.



(κωδικοποίηση της εντολής του παραδείγματος)

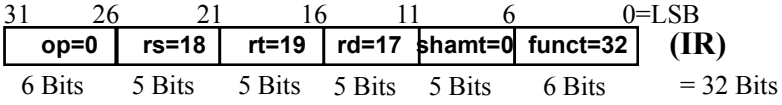
Αριθμητικές και Λογικές Εντολές (Τύπου R)

Εντολή	Περιγραφή
ADD $s1,$s2,$s3	Addition (προσημασμένοι αριθμοί - με υπερχείλιση)
ADDU $s1,$s2,$s3	Addition (μη προσημ/νοι αριθμοί - χωρίς υπερχείλιση)
SUB $s1,$s2,$s3	Subtract (προσημασμένοι αριθμοί - με υπερχείλιση)
SUBU $s1,$s2,$s3	Subtract (μη προσημ/νοι αριθμοί - χωρίς υπερχείλιση)
MULT $s2,$s3	Multiply (προσημασμένοι αριθμοί)
AND $s1,$s2,$s3	AND
OR $s1,$s2,$s3	OR
NOR $s1,$s2,$s3	NOR
XOR $s1,$s2,$s3	XOR

Αριθμητικές και Λογικές Εντολές (Τύπου R)

- `ADD rd, rs, rt  op=0x00  funct=0x20`
  - **Reg[rd] = Reg[rs] + Reg[rt]**
  - **Παράδειγμα :** `ADD $s1, $s2, $s3`

Αποθήκευση στον καταχωρητή $s1 = $17 του αθροίσματος των περιεχομένων των καταχωρητων $s2 = $18 και $s3 = $19, αντίστοιχα. Λαμβάνεται υπόψη η υπερχείλιση.

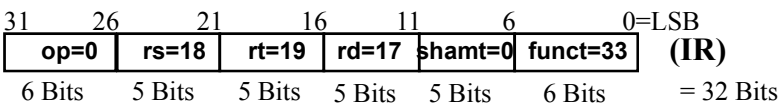


(κωδικοποίηση της εντολής του παραδείγματος)

Αριθμητικές και Λογικές Εντολές (Τύπου R)

- `ADDU rd, rs, rt  op=0x00  funct=0x21`
  - **Reg[rd] = Reg[rs] + Reg[rt]**
  - **Παράδειγμα :** `ADDU $s1, $s2, $s3`

Αποθήκευση στον καταχωρητή $s1 = $17 του αθροίσματος των περιεχομένων των καταχωρητων $s2 = $18 και $s3 = $19, αντίστοιχα. <u>Δεν</u> λαμβάνεται υπόψη η υπερχείλιση.

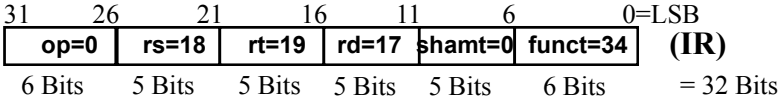


(κωδικοποίηση της εντολής του παραδείγματος)

Αριθμητικές και Λογικές Εντολές (Τύπου R)

- `SUB rd, rs, rt  op=0x00  funct=0x22`
  - **Reg[rd] = Reg[rs] - Reg[rt]**
  - **Παράδειγμα :** `SUB $s1, $s2, $s3`

Αποθήκευση στον καταχωρητή $s1 = $17 της διαφοράς των περιεχομένων των καταχωρητων $s2 = $18 και $s3 = $19, αντίστοιχα. Λαμβάνεται υπόψη η υπερχείλιση.

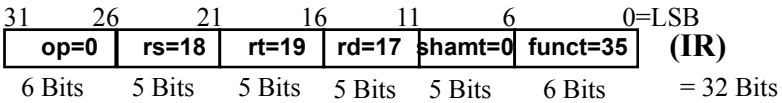


(κωδικοποίηση της εντολής του παραδείγματος)

Αριθμητικές και Λογικές Εντολές (Τύπου R)

- `SUBU rd, rs, rt  op=0x00  funct=0x23`
  - **Reg[rd] = Reg[rs] - Reg[rt]**
  - **Παράδειγμα :** `SUBU $s1, $s2, $s3`

Αποθήκευση στον καταχωρητή $s1 = $17 της διαφοράς των περιεχομένων των καταχωρητων $s2 = $18 και $s3 = $19, αντίστοιχα. <u>Δεν</u> λαμβάνεται υπόψη η υπερχείλιση.



(κωδικοποίηση της εντολής του παραδείγματος)

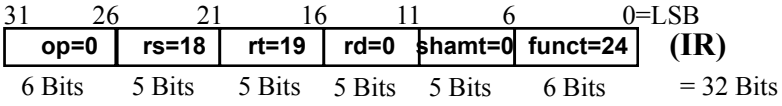
Αριθμητικές και Λογικές Εντολές (Τύπου R)

- `MULT rs, rt  op=0x00  funct=0x18`
  - **Hi : Lo = Reg[rs] x Reg[rt]**
  - **Παράδειγμα : `MULT $s2, $s3`**

Αποθήκευση στους επιπλέον καταχωρητές Hi και Lo του γινομένου (MSW και LSW, αντίστοιχα) των περιεχομένων των καταχωρητων $s2 = $18 επί $s3 = $19.

Η πράξη γίνεται μεταξύ προσημασμένων αριθμών αλλά δεν εξετάζεται η υπερχείλιση, που πρέπει να ελεγχθεί χωριστά μετά το τέλος της πράξης.

Hi = all 0's, εάν το γινόμενο που χωράει για αποθήκευση στον Lo είναι θετικός αριθμός.
Hi = all 1's, εάν το γινόμενο που χωράει για αποθήκευση στον Lo είναι αρνητικός αριθμός.

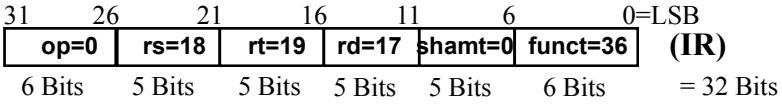


(κωδικοποίηση της εντολής του παραδείγματος)

Αριθμητικές και Λογικές Εντολές (Τύπου R)

- `AND rd, rs, rt  op=0x00  funct=0x24`
  - **Reg[rd] = Reg[rs] AND Reg[rt]**
  - **Παράδειγμα :** `AND $s1, $s2, $s3`

Αποθήκευση στον καταχωρητή $s1 = $17 της λογικής πράξης AND ανά ψηφίο των περιεχομένων των καταχωρητων $s2 = $18 και $s3 = $19, αντίστοιχα.

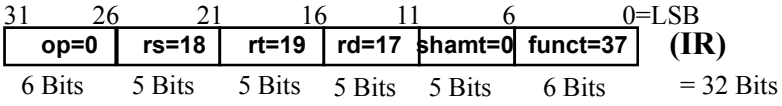


(κωδικοποίηση της εντολής του παραδείγματος)

Αριθμητικές και Λογικές Εντολές (Τύπου R)

- `OR rd, rs, rt  op=0x00  funct=0x25`
  - **Reg[rd] = Reg[rs] OR Reg[rt]**
  - **Παράδειγμα :** `OR $s1, $s2, $s3`

Αποθήκευση στον καταχωρητή $s1 = $17 της λογικής πράξης OR ανά ψηφίο των περιεχομένων των καταχωρητων $s2 = $18 και $s3 = $19, αντίστοιχα.

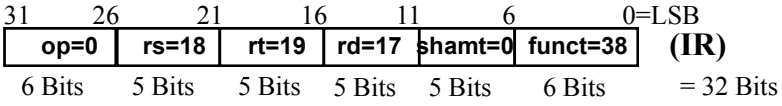


(κωδικοποίηση της εντολής του παραδείγματος)

Αριθμητικές και Λογικές Εντολές (Τύπου R)

- `XOR rd, rs, rt  op=0x00  funct=0x26`
  - **Reg[rd] = Reg[rs] XOR Reg[rt]**
  - **Παράδειγμα :** `XOR $s1, $s2, $s3`

Αποθήκευση στον καταχωρητή $s1 = $17 της λογικής πράξης XOR ανά ψηφίο των περιεχομένων των καταχωρητων $s2 = $18 και $s3 = $19, αντίστοιχα.

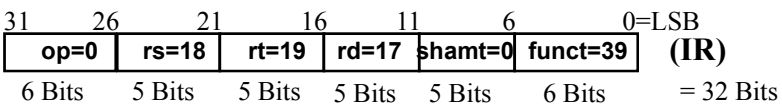


(κωδικοποίηση της εντολής του παραδείγματος)

Αριθμητικές και Λογικές Εντολές (Τύπου R)

- `NOR rd, rs, rt  op=0x00  funct=0x27`
  - **Reg[rd] = Reg[rs] NOR Reg[rt]**
  - **Παράδειγμα :** `NOR $s1, $s2, $s3`

Αποθήκευση στον καταχωρητή $s1 = $17 της λογικής πράξης NOR ανά ψηφίο των περιεχομένων των καταχωρητων $s2 = $18 και $s3 = $19, αντίστοιχα.



(κωδικοποίηση της εντολής του παραδείγματος)

Εντολές Ολίσθησης (Τύπου R)

Εντολή	Περιγραφή
SLL $s1,$s2, 3	Shift left logical (σταθερός αριθμός ολισθήσεων)
SRL $s1,$s2, 3	Shift right logical (σταθερός αριθμός ολισθήσεων)
SRA $s1,$s2, 3	Shift right arithmetic (σταθερός αριθμός ολισθήσεων με επέκταση πρόσημου)
SLLV $s1,$s2,$s3	Shift left logical variable (μεταβλητός αρ. ολισθήσεων)
SRLV $s1,$s2,$s3	Shift right logical variable (μεταβλητός αρ. ολισθήσεων)
SRAV $s1,$s2,$s3	Shift right arithmetic variable (μεταβλητός αριθμός ολισθήσεων με επέκταση πρόσημου)
NOP	No operation (κάνει μόνο καθυστέρηση) – Ψευδο-εντολή

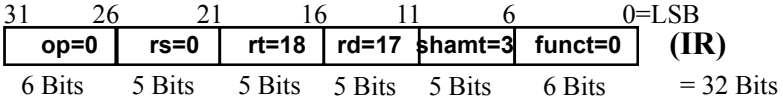
Η περιστροφή γίνεται με ψευδο-εντολές

Εντολές Ολίσθησης (Τύπου R)

- **`SLL rd, rt, shamt op=0x00 funct=0x00`**
  - **reg[rd] <= reg[rt] shifted left logical by shamt**
  - **Παράδειγμα : `SLL $s1, $s2, 3`**

Αριστερή λογική ολίσθηση του περιεχομένου του καταχωρητή $s2 = $18 κατά 3 (από 0 μέχρι 31) και αποθήκευση στον καταχωρητή $s1 = $17.

Η ψευδο-εντολή NOP μετατρέπεται στην εντολή SLL $zero, $zero, 0 με κωδικοποίηση όλα-0

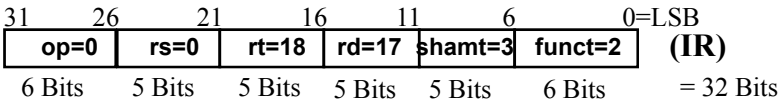


(κωδικοποίηση της εντολής του παραδείγματος)

Εντολές Ολίσθησης (Τύπου R)

- **`SRL rd, rt, shamt op=0x00 funct=0x02`**
  - **Reg[rd] = Reg[rt] shifted right logical by shamt**
  - **Παράδειγμα : `SRL $s1, $s2, 3`**

Δεξιά λογική ολίσθηση του περιεχομένου του καταχωρητή $s2 = $18 κατά 3 (από 0 μέχρι 31) και αποθήκευση στον καταχωρητή $s1 = $17.

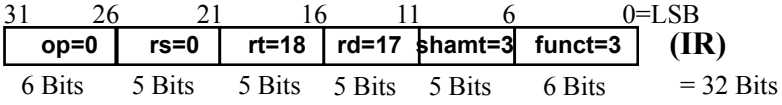


(κωδικοποίηση της εντολής του παραδείγματος)

Εντολές Ολίσθησης (Τύπου R)

- **`SRA rd, rt, shamt op=0x00 funct=0x03`**
  - **Reg[rd] = Reg[rt] shifted right arithmetic by shamt**
  - **Παράδειγμα : `SRA $s1, $s2, 3`**

Δεξιά αριθμητική ολίσθηση (που βασίζεται στο πρόσημο) του περιεχομένου του καταχωρητή $s2 = $18 κατά 3 (από 0 μέχρι 31) και αποθήκευση στον καταχωρητή $s1 = $17.

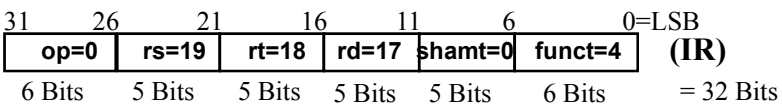


(κωδικοποίηση της εντολής του παραδείγματος)

Εντολές Ολίσθησης (Τύπου R)

- **`SLLV rd, rt, rs op=0x00 funct=0x04`**
  - **Reg[rd] = Reg[rt] shifted left logical by Reg[rs]**
  - **Παράδειγμα : `SLLV $s1, $s2, $s3`**

Μεταβλητή αριστερή λογική ολίσθηση του περιεχομένου του καταχωρητή $s2 = $18 κατά τόσα ψηφία (από 0 μέχρι 31) , όσα προσδιορίζονται στον καταχωρητή $s3 = $19, και αποθήκευση στον καταχωρητή $s1 = $17.

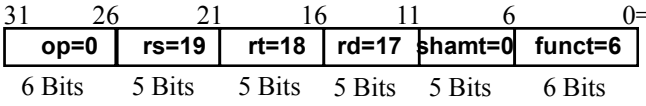


(κωδικοποίηση της εντολής του παραδείγματος)

Εντολές Ολίσθησης (Τύπου R)

- `SRLV rd, rt, rs  op=0x00 funct=0x06`
  - **Reg[rd] = Reg[rt] shifted right logical by Reg[rs]**
  - **Παράδειγμα :** `SRLV $s1, $s2, $s3`

Μεταβλητή δεξιά λογική ολίσθηση του περιεχομένου του καταχωρητή $s2 = $18 κατά τόσα ψηφία (από 0 μέχρι 31) , όσα προσδιορίζονται στον καταχωρητή $s3 = $19, και αποθήκευση στον καταχωρητή $s1 = $17.

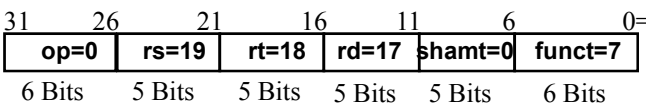


(κωδικοποίηση της εντολής του παραδείγματος)

Εντολές Ολίσθησης (Τύπου R)

- `SRAV rd, rt, rs  op=0x00 funct=0x07`
  - **Reg[rd] = Reg[rt] shifted right logical by Reg[rs]**
  - **Παράδειγμα :** `SRAV $s1, $s2, $s3`

Μεταβλητή δεξιά αριθμητική ολίσθηση του περιεχομένου του καταχωρητή $s2 = $18 κατά τόσα ψηφία (από 0 μέχρι 31) , όσα προσδιορίζονται στον καταχωρητή $s3 = $19, και αποθήκευση στον καταχωρητή $s1 = $17.



(κωδικοποίηση της εντολής του παραδείγματος)

Εντολές Ελέγχου Ροής Προγράμματος

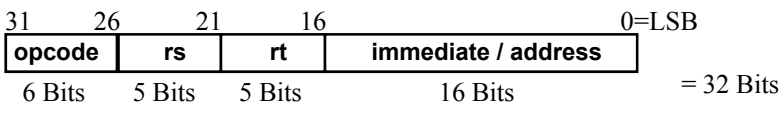
- **Εντολές διακλάδωσης με συνθήκη (conditional branch) - 85%**
- **Εντολές μεταπήδησης (Jump) - 5%**
- **Εντολές κλήσης διαδικασίας - επιστροφής από διαδικασία - 10%**

Εντολές Διακλάδωσης με Συνθήκη

- **Εντολές διακλάδωσης με συνθήκη (conditional branch)**
  - **ο μετρητής προγράμματος (program counter - PC) περιέχει τη διεύθυνση της εντολής που πρόκειται να ανακληθεί από τη μνήμη για να εκτελεσθεί μετά την τρέχουσα εντολή**
  - **Για να γίνει η αλλαγή στη ροή του προγράμματος εξετάζεται εάν πρώτα ικανοποιείται μία συγκεκριμένη συνθήκη, η συνθήκη διακλάδωσης**
  - **όταν δεν ικανοποιείται η συνθήκη, η επόμενη εντολή που πρόκειται να εκτελεσθεί είναι η αμέσως επόμενη εντολή μετά την εντολή διακλάδωσης (για τον MIPS ισχύει PC = PC + 4)**
  - **όταν ικανοποιείται η συνθήκη, η διεύθυνση της επόμενη εντολής που πρόκειται να εκτελεσθεί προσδιορίζεται με μετατόπιση (μ) που προστίθεται στον μετρητή προγράμματος (program counter - PC), που ήδη περιέχει τη διεύθυνση της αμέσως επόμενης εντολής μετά την εντολή διακλάδωσης (για τον MIPS ισχύει PC = PC + 4 + 4μ)**

PC-Σχετική Διευθυνσιοδότηση MIPS R2000

- **Μετατόπιση**
  - **η μετατόπιση είναι ένας προσημασμένος ακέραιος αριθμός που αντιστοιχεί στη διαφορά σε αριθμό εντολών (για MIPS) της εντολής που θα εκτελεσθεί όταν ικανοποιείται η συνθήκη, από την εντολή που έπεται της εντολής διακλάδωσης.**
  - **συμπεριλαμβάνεται στην εντολή διακλάδωσης υπό συνθήκη σαν άμεση τιμή στο πεδίο immediate.**
- **PC-Σχετική Διευθυνσιοδότηση (PC - relative addressing)**
  - **ονομάζεται ο τρόπος διευθυνσιοδότησης της μνήμης εντολών που χρησιμοποιείται στις εντολές διακλάδωσης υπό συνθήκη που προσδιορίζουν άμεσα με μετατόπιση τη διεύθυνση της εντολής, που θα εκτελεσθεί όταν ικανοποιείται η συνθήκη.**



PC-Σχετική Διευθυνσιοδότηση

- **Πότε χρησιμοποιείται ;**
  - **για να χρησιμοποιηθεί η PC - σχετική διευθυνσιοδότηση πρέπει η μετατόπιση να είναι γνωστή κατά το στάδιο της μεταγλώττισης, οπότε έχουμε ανεξαρτησία θέσης (δηλαδή, δεν εξαρτάται η μετατόπιση από το που θα φορτωθεί το πρόγραμμα στη μνήμη κατά την εκτέλεση του).**
  - **όταν η μετατόπιση δεν είναι γνωστή κατά το στάδιο της μεταγλώττισης, τότε η διεύθυνση της εντολής, που θα εκτελεσθεί όταν ικανοποιείται η συνθήκη, προσδιορίζεται δυναμικά κατά την εκτέλεση του προγράμματος συνήθως με διευθυνσιοδότηση καταχωρητή, που περιέχει τη διεύθυνση αυτής της εντολής, χωρίς να αποκλείεται η χρήση κάποιου άλλου τρόπου διευθυνσιοδότησης.**

PC-Σχετική Διευθυνσιοδότηση MIPS R2000

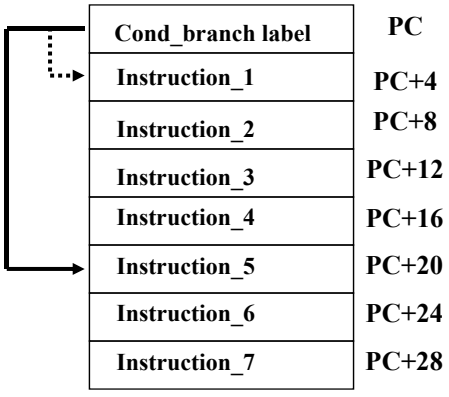
- **Εντολές διακλάδωσης υπό συνθήκη (conditional branch) - 85%**

Πρόγραμμα με PC - σχετική διευθυν/ση

```
          ……..
          Cond_branch label
          Instruction_1
          Instruction_2
          Instruction_3
          Instruction_4
label:    Instruction_5
          Instruction_6
          Instruction_7
          ……..
```

Η εντολή Cond_branch αλλάζει τη ροή του προγράμματος. Όταν ικανοποιείται η συνθήκη, η αμέσως επόμενη εντολή που θα εκτελεσθεί είναι η Instruction_5, που προσδιορίζεται από το label, αλλιώς είναι η Instruction_1

Μνήμη Εντολών των 4 bytes (32 bits)



Η μετατόπιση είναι η διαφορά των εντολών της Instruction_5 από την Instruction_1, δηλαδή +4

Εντολές Σύγκρισης - Set (Τύπου Ι)

Εντολή	Περιγραφή
SLTI $t0, $s0, 10	Set less than immediate If Reg[s0] < 10 then Reg[t0]=1 else Reg[t0]=0 Η σύγκριση γίνεται μεταξύ προσημασμένων αριθμών
SLTIU $t0, $s0, 10	Set less than unsigned immediate If Reg[s0] < 10 then Reg[t0]=1 else Reg[t0]=0 Η σύγκριση γίνεται μεταξύ μη προσημασμένων αριθμών

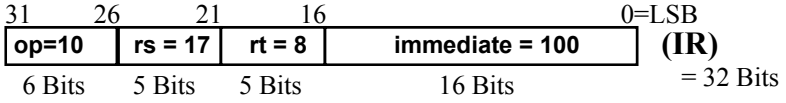
Δεν υπάρχει έλεγχος υπερχείλισης. Το αποτέλεσμα της σύγκρισης είναι πάντα ορθό στην περίπτωση που οι συγκρινόμενοι προσημασμένοι αριθμοί ανήκουν στην περιοχή από 2^{30}-1 (0011 1111 1111 1111 1111 1111 1111 1111) μέχρι - 2^{30} (1100 0000 0000 0000 0000 0000 0000 0000). Ισχύει και για μη προσημασμένους αριθμούς που ανήκουν στην περιοχή από 2^{31}-1 μέχρι 0.

Η υπόλοιπες εντολές σύγκρισης είναι ψευδο-εντολές

Εντολές Σύγκρισης - Set (Τύπου I)

- `SLTI rt, rs, immediate op=0x0A`
  - **IF** **Reg[rs] < sign_ext(immediate)**
    THEN **Reg[rt] = 1**
    ELSE **Reg[rt] = 0**
  - **Παράδειγμα :** `SLTI $t0, $s1, 100`

Εάν το περιεχόμενο του καταχωρητή $s1 = $17 είναι μικρότερο από το περιεχόμενο του πεδίου immediate (100), αφού έχει υποστεί επέκταση πρόσημου, τότε στον καταχωρητή $t0 = $8 αποθηκεύεται το 1, αλλιώς αποθηκεύεται το 0. Η σύγκριση γίνεται μεταξύ προσημασμένων αριθμών.

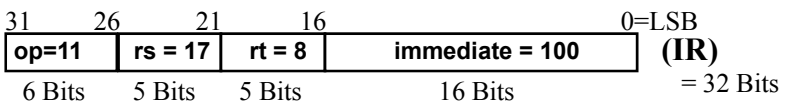


(κωδικοποίηση της εντολής του παραδείγματος)

Εντολές Σύγκρισης - Set (Τύπου I)

- `SLTIU rt, rs, immediate op=0x0B`
  - **IF** **Reg[rs] < sign_ext(immediate)**
    THEN **Reg[rt] = 1**
    ELSE **Reg[rt] = 0**
  - **Παράδειγμα :** `SLTIU $t0, $s1, 100`

Εάν το περιεχόμενο του καταχωρητή $s1 = $17 είναι μικρότερο από το περιεχόμενο του πεδίου immediate (100), αφού έχει υποστεί επέκταση πρόσημου, τότε στον καταχωρητή $t0 = $8 αποθηκεύεται το 1, αλλιώς αποθηκεύεται το 0. Η σύγκριση γίνεται μεταξύ μη προσημασμένων αριθμών.



(κωδικοποίηση της εντολής του παραδείγματος)

Εντολές Σύγκρισης - Set (Τύπου R)

Εντολή	Περιγραφή
SLT $t0, $s0, $s1	Set less than If Reg[s0] < Reg[s1] then Reg[t0]=1 else Reg[t0]=0 Η σύγκριση γίνεται μεταξύ προσημασμένων αριθμών
SLTU $t0, $s0, $s1	Set less than unsigned If Reg[s0] < Reg[s1] then Reg[t0]=1 else Reg[t0]=0 Η σύγκριση γίνεται μεταξύ μη προσημασμένων αριθμών

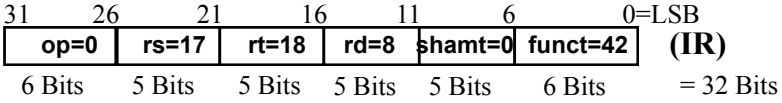
Δεν υπάρχει έλεγχος υπερχείλισης. Το αποτέλεσμα της σύγκρισης είναι πάντα ορθό στην περίπτωση που οι συγκρινόμενοι προσημασμένοι αριθμοί ανήκουν στην περιοχή από 2^{30}-1 (0011 1111 1111 1111 1111 1111 1111 1111) μέχρι - 2^{30} (1100 0000 0000 0000 0000 0000 0000 0000). Ισχύει και για μη προσημασμένους αριθμούς που ανήκουν στην περιοχή από 2^{31}-1 μέχρι 0.

Η υπόλοιπες εντολές σύγκρισης είναι ψευδο-εντολές

Εντολές Σύγκρισης - Set (Τύπου R)

- `SLT rd, rs, rt  op=0x00 funct=0x2A`
  - **IF** **Reg[rs] < Reg[rt]**
    THEN **Reg[rd] = 1**
    ELSE **Reg[rd] = 0**
  - **Παράδειγμα :** `SLT $t0, $s1, $s2`

Εάν το περιεχόμενο του καταχωρητή $s1 = $17 είναι μικρότερο από το περιεχόμενο του καταχωρητή $s2 = $18 , τότε στον καταχωρητή $t0 = $8 αποθηκεύεται το 1, αλλιώς αποθηκεύεται το 0. Η σύγκριση γίνεται μεταξύ προσημασμένων αριθμών.

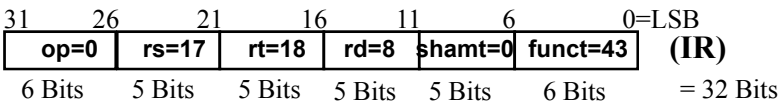


(κωδικοποίηση της εντολής του παραδείγματος)

Εντολές Σύγκρισης - Set (Τύπου R)

- `SLTU rd, rs, rt  op=0x00 funct=0x2B`
  - **IF** **Reg[rs] < Reg[rt]**
    THEN **Reg[rd] = 1**
    ELSE **Reg[rd] = 0**
  - **Παράδειγμα :** `SLTU $t0, $s1, $s2`

Εάν το περιεχόμενο του καταχωρητή $s1 = $17 είναι μικρότερο από το περιεχόμενο του καταχωρητή $s2 = $18 , τότε στον καταχωρητή $t0 = $8 αποθηκεύεται το 1, αλλιώς αποθηκεύεται το 0. Η σύγκριση γίνεται μεταξύ μη προσημασμένων αριθμών.



(κωδικοποίηση της εντολής του παραδείγματος)

Εντολές Διακλάδωσης (Τύπου Ι)

Εντολή	Περιγραφή
BEQ $s1, $s2, label	Branch on equal If reg[s1] = reg[s2] then PC <= PC + 4 + 4μ else PC <= PC + 4
BNE $s1, $s2, label	Branch on not equal If reg[s1] ≠ reg[s2] then PC <= PC + 4 + 4μ else PC <= PC + 4

Ο MIPS υλοποιεί και άλλες εντολές διακλάδωσης, ενώ υπάρχουν και ψευδο-εντολές

PC - σχετική διευθυνσιοδότηση: Το label προσδιορίζει την εντολή, που θα εκτελεσθεί όταν ικανοποιείται η συνθήκη, και μετατρέπεται σε μετατόπιση (μ) κατά το στάδιο της μεταγλώττισης

Εντολές Διακλάδωσης (Τύπου Ι)

- Στους πραγματικούς επεξεργαστές MIPS, οι εντολές διακλάδωσης είναι καθυστερημένες διακλαδώσεις (delayed branches)
- Πάντα εκτελείται η εντολή που ακολουθεί τη διακλάδωση, που βρίσκεται στη διεύθυνση PC+4 και ονομάζεται delay slot
- Μετά την εκτέλεση της delay slot, εάν ικανοποιείται η συνθήκη της διακλάδωσης, εκτελείται η εντολή που βρίσκεται στη διεύθυνση PC+4+4μ
- Οι καθυστερημένες διακλαδώσεις επηρεάζουν τον υπολογισμό της μετατόπισης (μ) που γίνεται με βάση το PC+4 και όχι το PC

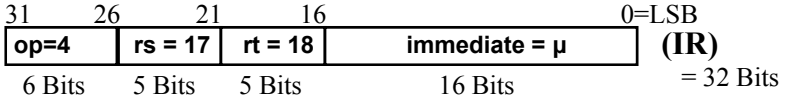
Προσοχή γιατί ο SPIM συνήθως δεν προσομοιώνει την καθυστερημένη διακλάδωση και υπολογίζει τη σχετική απόσταση με βάση το PC

Προσοχή, δεν θα υλοποιήσουμε καθυστερημένες διακλαδώσεις, αν και θα υπολογίσουμε τη σχετική απόσταση με βάση το PC+4

Εντολές Διακλάδωσης (Τύπου Ι)

- `BEQ rs, rt, label  op=0x04`
  - **IF** reg[rs] = reg[rt]
    THEN PC <= PC + 4 + 4μ — Διεύθυνση προορισμού διακλάδωσης
    ELSE PC <= PC + 4 — Διεύθυνση επόμενης εντολής
  - **Παράδειγμα :** `BEQ $s1, $s2, label`

Εάν το περιεχόμενο του καταχωρητή $s1 = $17 είναι ίσο με το περιεχόμενο του καταχωρητή $s2 = $18, τότε PC <= PC+4+4μ, αλλιώς PC <= PC+4. Κατά τη μεταγλώττιση αντικαθίσταται το label με την μετατόπιση, που είναι η διαφορά σε αριθμό εντολών της εντολής που προσδιορίζει το label και της εντολής που έπεται της εντολής διακλάδωσης (delay slot). Η μετατόπιση κυμαίνεται από -2^{15} μέχρι $2^{15}-1$.



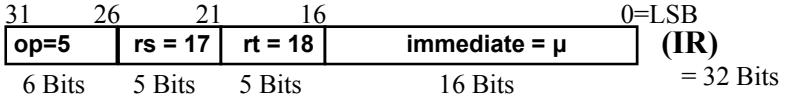
(IR) = 32 Bits

(κωδικοποίηση της εντολής του παραδείγματος)

Εντολές Διακλάδωσης (Τύπου Ι)

- `BNE rs, rt, label  op=0x05`
  - **IF** reg[rs] ≠ reg[rt]
    THEN PC <= PC + 4 + 4μ — Διεύθυνση προορισμού διακλάδωσης
    ELSE PC <= PC + 4 — Διεύθυνση επόμενης εντολής
  - **Παράδειγμα :** `BNE $s1, $s2, label`

Εάν το περιεχόμενο του καταχωρητή $s1 = $17 είναι άνισο με το περιεχόμενο του καταχωρητή $s2 = $18, τότε PC <= PC+4+4μ, αλλιώς PC <= PC+4. Κατά τη μεταγλώττιση αντικαθίσταται το label με την μετατόπιση, που είναι η διαφορά σε αριθμό εντολών της εντολής που προσδιορίζει το label και της εντολής που έπεται της εντολής διακλάδωσης (delay slot). Η μετατόπιση κυμαίνεται από -2^{15} μέχρι $2^{15}-1$.



(IR) = 32 Bits

(κωδικοποίηση της εντολής του παραδείγματος)

Εντολές Διακλάδωσης με Συνθήκη

◆ **Υλοποίηση συνθηκών προσημασμένης σύγκρισης**

- **if reg[s1] = reg[s2] go to label**
  - `BEQ $s1, $s2, label`
- **if reg[s1] ≠ reg[s2] go to label**
  - `BNE $s1, $s2, label`
- **if reg[s1] < reg[s2] go to label**
  - `SLT $t0, $s1, $s2`
  - `BNE $t0, $zero, label`
- **if reg[s1] ≥ reg[s2] go to label**
  - `SLT $t0, $s1, $s2`
  - `BEQ $t0, $zero, label`
- **if reg[s1] > reg[s2] go to label**
  - `SLT $t0, $s2, $s1`
  - `BNE $t0, $zero, label`
- **if reg[s1] ≤ reg[s2] go to label**
  - `SLT $t0, $s2, $s1`
  - `BEQ $t0, $zero, label`

Δεν υπάρχει έλεγχος υπερχείλισης. Το αποτέλεσμα της σύγκρισης (<, ≤, ≥, >) είναι πάντα ορθό στην περίπτωση που οι συγκρινόμενοι προσημασμένοι αριθμοί ανήκουν στην περιοχή από 2^{30}-1 (0011 1111 1111 1111 1111 1111 1111 1111) μέχρι - 2^{30} (1100 0000 0000 0000 0000 0000 0000 0000)

Άλλες Εντολές Διακλάδωσης (Τύπου Ι)

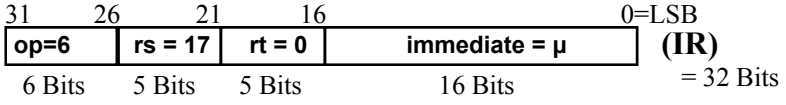
Εντολή	Περιγραφή
BLEZ $s1, label	Branch on less than equal zero If Reg[s1] $\leq$ 0 then PC = PC + 4 + 4μ else PC = PC + 4
BGTZ $s1, label	Branch on greater than zero If Reg[s1] $>$ 0 then PC = PC + 4 + 4μ else PC = PC + 4
BLTZ $s1, label	Branch on less than zero If Reg[s1] $<$ 0 then PC = PC + 4 + 4μ else PC = PC + 4
BGEZ $s1, label	Branch on greater than equal zero If Reg[s1] $\geq$ 0 then PC = PC + 4 + 4μ else PC = PC + 4

PC - σχετική διευθυνσιοδότηση

Εντολές Διακλάδωσης (Τύπου Ι)

- `BLEZ rs, label  op=0x06`
  - **IF** **Reg[rs] ≤ 0**
    THEN **PC = PC + 4 + 4μ**
    ELSE **PC = PC + 4**
  - **Παράδειγμα :** `BLEZ $s1, label`

Εάν το περιεχόμενο του καταχωρητή $s1 = $17 είναι μικρότερο ή ίσο του μηδενός, τότε PC = PC+4+4μ, αλλιώς PC = PC+4.
Κατά τη μεταγλώττιση αντικαθίσταται το label με την μετατόπιση, που είναι η διαφορά σε αριθμό εντολών της εντολής που προσδιορίζει το label και της εντολής που έπεται της εντολής διακλάδωσης.
Η μετατόπιση κυμαίνεται από -2^{15} μέχρι $2^{15}-1$.

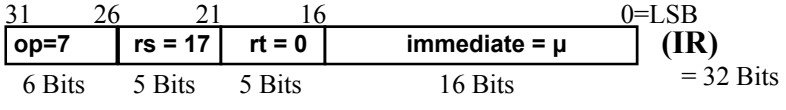


(κωδικοποίηση της εντολής του παραδείγματος)

Εντολές Διακλάδωσης (Τύπου Ι)

- `BGTZ rs, label  op=0x07`
  - **IF** **Reg[rs] > 0**
    THEN **PC = PC + 4 + 4μ**
    ELSE **PC = PC + 4**
  - **Παράδειγμα :** `BGTZ $s1, label`

Εάν το περιεχόμενο του καταχωρητή $s1 = $17 είναι μεγαλύτερο του μηδενός, τότε PC = PC+4+4μ, αλλιώς PC = PC+4.
Κατά τη μεταγλώττιση αντικαθίσταται το label με την μετατόπιση, που είναι η διαφορά σε αριθμό εντολών της εντολής που προσδιορίζει το label και της εντολής που έπεται της εντολής διακλάδωσης.
Η μετατόπιση κυμαίνεται από -2^{15} μέχρι $2^{15}-1$.



(κωδικοποίηση της εντολής του παραδείγματος)

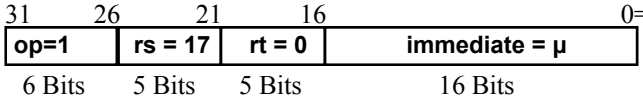
Εντολές Διακλάδωσης (Τύπου I)

- `BLTZ rs, label  op=0x01  rt = 0x00`
  - **IF** **Reg[rs] < 0**
    THEN **PC = PC + 4 + 4μ**
    ELSE **PC = PC + 4**

Επέκταση opcode με τη χρήση του πεδίου rt

  - **Παράδειγμα :** `BLTZ $s1, label`

Εάν το περιεχόμενο του καταχωρητή $s1 = $17 είναι μικρότερο του μηδενός, τότε PC = PC+4+4μ, αλλιώς PC = PC+4.
Κατά τη μεταγλώττιση αντικαθίσταται το label με την μετατόπιση, που είναι η διαφορά σε αριθμό εντολών της εντολής που προσδιορίζει το label και της εντολής που έπεται της εντολής διακλάδωσης.
Η μετατόπιση κυμαίνεται από -2^{15} μέχρι $2^{15}-1$.



(κωδικοποίηση της εντολής του παραδείγματος)

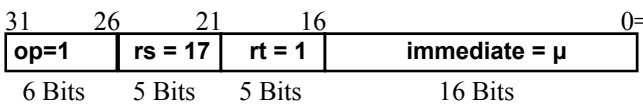
Εντολές Διακλάδωσης (Τύπου I)

- `BGEZ rs, label  op=0x01  rt = 0x01`
  - **IF** **Reg[rs] ≥ 0**
    THEN **PC = PC + 4 + 4μ**
    ELSE **PC = PC + 4**

Επέκταση opcode με τη χρήση του πεδίου rt

  - **Παράδειγμα :** `BGEZ $s1, label`

Εάν το περιεχόμενο του καταχωρητή $s1 = $17 είναι μεγαλύτερο ή ίσο του μηδενός, τότε PC = PC+4+4μ, αλλιώς PC = PC+4.
Κατά τη μεταγλώττιση αντικαθίσταται το label με την μετατόπιση, που είναι η διαφορά σε αριθμό εντολών της εντολής που προσδιορίζει το label και της εντολής που έπεται της εντολής διακλάδωσης.
Η μετατόπιση κυμαίνεται από -2^{15} μέχρι $2^{15}-1$.



(κωδικοποίηση της εντολής του παραδείγματος)

Εντολές Ελέγχου Ροής Προγράμματος

- **Εντολές μεταπήδησης (Jump) - 5%**
  - **η αλλαγή στη ροή του προγράμματος γίνεται χωρίς συνθήκη**
  - **για τον προσδιορισμό της εντολής που θα εκτελεσθεί μετά τη μεταπήδηση χρησιμοποιείται:**
    - **διευθυνσιοδότηση καταχωρητή**
    - **ψευδο-άμεση διευθυνσιοδότηση (για MIPS R2000) εάν η διεύθυνση της επόμενης εντολής που πρόκειται να εκτελεσθεί, εάν είναι γνωστή κατά το στάδιο της μεταγλώττισης**

Εντολές Μεταπήδησης (Τύπου R)

Εντολή	Περιγραφή
JR $t0	Jump register, PC = Reg[t0]

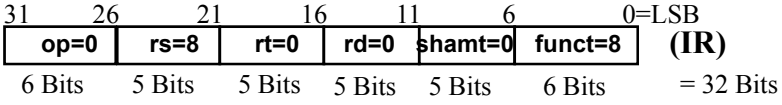
Διευθυνσιοδότηση καταχωρητή

Εντολές Μεταπήδησης (Τύπου R)

- `JR rs` `op=0x00 funct=0x08`
  - **PC = Reg[rs]**

  Παράδειγμα : `JR $t0`

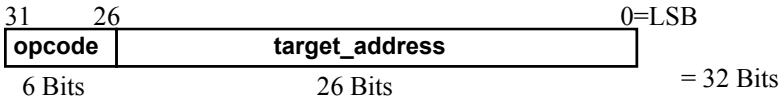
Η ενεργή διεύθυνση της εντολής που πρόκειται να εκτελεσθεί μετά τη μεταπήδηση είναι αποθηκευμένη αρχικά στον καταχωρητή $t0 = $8 και μεταφέρεται στο μετρητή προγράμματος PC.



(κωδικοποίηση της εντολής του παραδείγματος)

Ψευδο-Άμεση Διευθυνσιοδότηση MIPS R2000

- **Προσδιορισμός ενεργούς διεύθυνσης**
  - **Η ενεργός διεύθυνση της επόμενης εντολής που θα εκτελεστεί λόγω της μεταπήδησης προσδιορίζεται ως ακολούθως:**
    - **τα 2 λιγότερο σημαντικά ψηφία της διεύθυνσης είναι 0 λόγω ευθυγράμμισης (εντολή 4 bytes)**
    - **τα 26 επόμενα λιγότερο σημαντικά ψηφία της διεύθυνσης προσδιορίζονται από το πεδίο target_address**
    - **τα 4 περισσότερο σημαντικά ψηφία της διεύθυνσης παραμένουν ως έχουν (όριο τα 256 MB = 64 M εντολές)**



Εντολές Μεταπήδησης (Τύπου J)

Εντολή	Περιγραφή
J target	Jump, PC = PC(4msb):target_address:00

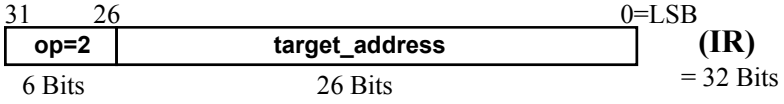
Ψευδο-άμεση διευθυνσιοδότηση

Εντολές Μεταπήδησης (Τύπου J)

- `J target op=0x02`
  - **PC = PC(4msb) : target_address : 00**

    Παράδειγμα : `J target`

Η ενεργή διεύθυνση της εντολής που πρόκειται να εκτελεσθεί μετά τη μεταπήδηση προσδιορίζεται σύμφωνα με τη ψευδο-άμεση διευθυνσιοδότηση και μεταφέρεται στο μετρητή προγράμματος PC.
Η target_address προσδιορίζεται από το μεταγλωττιστή.



(κωδικοποίηση της εντολής του παραδείγματος)

Εντολές Ελέγχου Ροής Προγράμματος

◆ **Εντολές κλήσης διαδικασίας - επιστροφής από διαδικασία - 10%**

- **με την κλήση της διαδικασίας η επόμενη εντολή που πρόκειται να εκτελεσθεί είναι η πρώτη εντολή της διαδικασίας που καλείται.**
- **για τον προσδιορισμό της διεύθυνσης της πρώτης εντολής της διαδικασίας που καλείται, εάν είναι γνωστή κατά το στάδιο της μεταγλώττισης χρησιμοποιείται ψευδο-άμεση διευθυνσιοδότηση (για MIPS), αλλιώς προσδιορίζεται δυναμικά κατά την εκτέλεση του προγράμματος με διευθυνσιοδότηση καταχωρητή.**
- **μετά την ολοκλήρωση της εκτέλεσης της διαδικασίας η ροή του προγράμματος επιστρέφει στην αμέσως επόμενη εντολή μετά την εντολή κλήσης της διαδικασίας που καλεί (με την εκτέλεση της εντολής επιστροφής σαν τελευταία εντολή της διαδικασίας που καλείται).**
- **για να εκτελεσθεί η εντολή επιστροφής πρέπει η διεύθυνσή της αμέσως επόμενης εντολής μετά την εντολή κλήσης της διαδικασίας να έχει ήδη αποθηκευθεί στον καταχωρητή $ra ή στη στοίβα κατά την εκτέλεση της εντολής κλήσης της διαδικασίας.**

Εντολές Ελέγχου Ροής Προγράμματος

◆ **Εντολές κλήσης διαδικασίας - επιστροφής από διαδικασία - 10%**

- **Μέθοδοι διατήρησης περιεχομένων καταχωρητών για να χρησιμοποιηθούν ξανά από τη διαδικασία που καλεί μετά την εκτέλεση της διαδικασίας που καλείται**
  - **η διαδικασία που καλεί σώζει τα περιεχόμενα των καταχωρητών, που θέλει να διατηρήσει και μετά την εκτέλεση της διαδικασίας που καλείται, σε μία στοίβα πριν από την κλήση της διαδικασίας που καλείται (caller - saving)**
  - **η διαδικασία που καλείται σώζει τα περιεχόμενα των καταχωρητών που θα χρησιμοποιήσει κατά την εκτέλεσή της σε μία στοίβα (callee -saving)**
- **Μεταφορά παραμέτρων από τη μία διαδικασία στην άλλη γίνεται μέσω καταχωρητών ($a0-$a3, $v0, $v1) και μέσω της στοίβας.**

Η στοίβα είναι πολύ χρήσιμη και υλοποιείται στη μνήμη σαν stack segment

Εντολές Κλήσης Διαδικασίας (Τύπου R)

Εντολή	Περιγραφή
JALR $t0	Jump and link register, Reg[ra] = PC+4 & PC = Reg[t0]

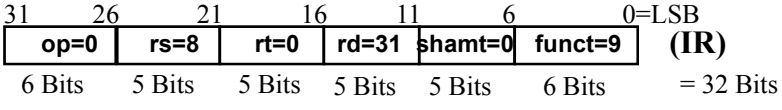
Διευθυνσιοδότηση καταχωρητή

Εντολές Κλήσης Διαδικασίας (Τύπου R)

- `JALR rs, rd` `op=0x00 funct=0x09`
  - **Reg[rd] = PC + 4 (default : $ra = $31)**
  - **PC = Reg[rs]**

Παράδειγμα : `JALR $t0`

Η διεύθυνση της επόμενης εντολής (PC+4) αποθηκεύεται στον $ra = $31 κατά τη διαδικασία της σύνδεσης (link).
Η ενεργή διεύθυνση της εντολής που πρόκειται να εκτελεσθεί μετά τη μεταπήδηση είναι αποθηκευμένη αρχικά στον καταχωρητή $t0 = $8 και μεταφέρεται στο μετρητή προγράμματος PC.



(κωδικοποίηση της εντολής του παραδείγματος)

Εντολές Κλήσης Διαδικασίας (Τύπου J)

Εντολή	Περιγραφή
JAL target	Jump and link, Reg[ra] = PC+4 & PC = PC(4msb):target_address:00

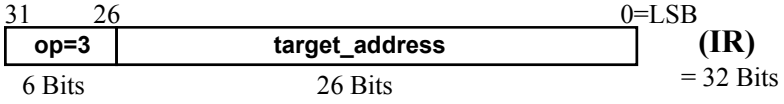
Ψευδο-άμεση διευθυνσιοδότηση

Εντολές Κλήσης Διαδικασίας (Τύπου J)

- `JAL target op=0x03`
  - **Reg[ra] = PC + 4**
  - **PC = PC(4msb) : target_address : 00**

Παράδειγμα : `JAL target`

Η διεύθυνση της επόμενης εντολής (PC+4) αποθηκεύεται στον $ra = $31 κατά τη διαδικασία της σύνδεσης (link).
Η ενεργή διεύθυνση της εντολής που πρόκειται να εκτελεσθεί μετά τη μεταπήδηση προσδιορίζεται σύμφωνα με τη ψευδο-άμεση διευθυνσιοδότηση και μεταφέρεται στο μετρητή προγράμματος PC.
Η target_address προσδιορίζεται από το μεταγλωττιστή.



(κωδικοποίηση της εντολής του παραδείγματος)

Εντολές Επιστροφής από Διαδικασία (Τύπου R)

Εντολή	Περιγραφή
JR $ra	Jump register, PC = Reg[ra]

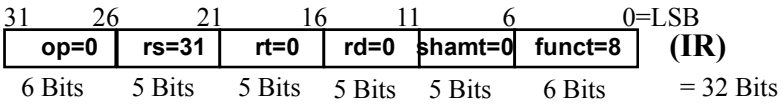
Διευθυνσιοδότηση καταχωρητή

Εντολές Επιστροφής από Διαδικασία (Τύπου R)

- `JR rs` `op=0x00 funct=0x08`
  - **PC = Reg[rs]**

    Παράδειγμα : `JR $ra`

Η ενεργή διεύθυνση της εντολής που πρόκειται να εκτελεσθεί μετά την επιστροφή από τη διαδικασία αποθηκεύεται στον καταχωρητή $ra = $31 κατά την κλήση της διαδικασίας (με τη χρήση των εντολών JAL και JALR) και μεταφέρεται στο μετρητή προγράμματος PC.



(κωδικοποίηση της εντολής του παραδείγματος)

Εντολές Κλήσης Διαδικασίας με Συνθήκη (Τύπου Ι)

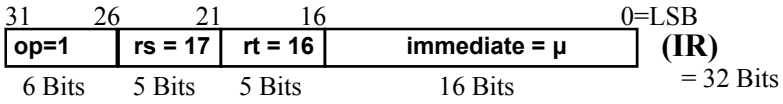
Εντολή	Περιγραφή
BLTZAL $s1, label	Branch on less than zero and link Reg[ra] = PC+4 & If Reg[s1] $<$ 0 then PC = PC + 4 + 4μ else PC = PC + 4
BGEZAL $s1, label	Branch on greater than equal zero and link Reg[ra] = PC+4 & If Reg[s1] $\geq$ 0 then PC = PC + 4 + 4μ else PC = PC + 4

PC-σχετική διευθυνσιοδότηση

Εντολές Κλήσης Διαδικασίας με Συνθήκη (Τύπου Ι)

- **`BLTZAL rs, label  op=0x01  rt = 0x10`**
  - **Reg[ra] = PC + 4** — **Επέκταση opcode με τη χρήση του πεδίου rt**
  - **IF Reg[rs] < 0 THEN PC = PC + 4 + 4μ ELSE PC = PC + 4**
  - **Παράδειγμα : `BLTZAL $s1, label`**

Η διεύθυνση της επόμενης εντολής (PC+4) αποθηκεύεται στον $ra = $31 κατά τη διαδικασία της σύνδεσης (link).
Εάν το περιεχόμενο του καταχωρητή $s1 = $17 είναι μικρότερο του μηδενός, τότε PC = PC+4+4μ, αλλιώς PC = PC+4.
Κατά τη μεταγλώττιση αντικαθίσταται το label με την μετατόπιση, που είναι η διαφορά σε αριθμό εντολών της εντολής που προσδιορίζει το label και της εντολής που έπεται της εντολής διακλάδωσης.
Η μετατόπιση κυμαίνεται από -2^{15} μέχρι $2^{15}-1$.



Εντολές Κλήσης Διαδικασίας με Συνθήκη (Τύπου Ι)

- **`BGEZAL rs, label  op=0x01  rt = 0x11`**
  - **Reg[ra] = PC + 4** — **Επέκταση opcode με τη χρήση του πεδίου rt**
  - **IF Reg[rs] ≥ 0 THEN PC = PC + 4 + 4μ ELSE PC = PC + 4**
  - **Παράδειγμα : `BGEZAL $s1, label`**

Η διεύθυνση της επόμενης εντολής (PC+4) αποθηκεύεται στον $ra = $31 κατά τη διαδικασία της σύνδεσης (link).
Εάν το περιεχόμενο του καταχωρητή $s1 = $17 είναι μεγαλύτερο ή ίσο του μηδενός, τότε PC = PC+4+4μ, αλλιώς PC = PC+4.
Κατά τη μεταγλώττιση αντικαθίσταται το label με την μετατόπιση, που είναι η διαφορά σε αριθμό εντολών της εντολής που προσδιορίζει το label και της εντολής που έπεται της εντολής διακλάδωσης.
Η μετατόπιση κυμαίνεται από -2^{15} μέχρι $2^{15}-1$.