ΗΥ425 - Αρχιτεκτονική Υπολογιστών - Μ2

Χ. Σωτηρίου

24 Σεπτεμβρίου 2001

Τύποι Συνόλων Εντολών

- ο τρόπος εσωτερικής αποθήκευσης στο CPU είναι η πιο βασική διαφορά ανάμεσα σε τύπους συνόλων εντολών.

Σωρό (*Stack*)	Συσσωρευτή (*Accumulator*)	Καταχωρητή (καταχωρητής - μνήμη)	Καταχωρητή (φόρτωσης - αποκήκευσης)
`Push A`	`Load A`	`Load R1, A`	`Load R1, A`
`Push B`	`Add B`	`Add R1, B`	`Load R2, B`
`Add`	`Store C`	`Store C, R1`	`Add R3, R1, R2`
`Pop C`			`Store C, R3`

- μετά το 1980 σχεδόν κάθε μηχάνημα είναι τύπου φόρτωσης - αποκήκευσης καταχωρητών ($load - store\ register$).

- λέγονται και '$General - Purpose - Registers$' ($GPL$).

Αρχιτεκτονικές με Γενικούς Καταχωρητές

- ευκολότεροι στην χρήση για τον μεταγλωτιστή (*compiler*).
- μειώνουν την κίνηση μεταξύ CPU και μνήμης.
- βελτιώνουν την πυκνότητα των εντολών λόγω κωδικοποίησης των καταχωρητών.
- επιτρέπουν την εκμετάλλευση περισσότερου παραλληλισμού.
- υποδιαρούνται ανάλογα με δύο χαρακτηριστικά:
- τον αριθμό των δρώμενων σε μια εντολή ALU.
- πόσα απο τα δρώμενα μιας εντολής ALU είναι διευθύνσεις μνήμης.

Αρχιτεκτονικές με Γενικούς Καταχωρητές

Αριθμός Διευθύνσεων Μνήμης	Μέγιστος Αριθμός Δρώμενων	Παραδείγματα
0	3	$SPARC$, $MIPS$, $PowerPC$, $Alpha$
1	2	$80x86$, $680x0$
2	2	VAX
3	3	VAX

Αρχιτεκτονικές με Γενικούς Καταχωρητές

Τύπος	Πλεονεκτήματα	Μειονεκτήματα
Καταχωρητή - καταχωρητή (0, 3)	Απλή, σταθερός κωδικοποιητής. Ο χρόνος εκτέλεσης εντολών μοιάζει.	Απαιτούν μεγαλύτερο αριθμό εντολών από μηχανές που επιτρέπουν πρόσβαση μνήμης στις εντολές.
Καταχωρητή - μνήμης (1, 2)	Πρόσβαση στα δεδομένα μνήμης χωρίς φόρτωμα. Εύκολη κωδικοποίηση και καλή πυκνότητα κώδικα.	Το πρώτο από τα δύο ή τρία πάντα μεταβάλλεται. Καθιστά δυσνόητη τη σειρά εκτέλεσης των εντολών στην επιλογή πιθανών αριθμό διαφορετικών χρόνων CPI, με τα δεδομένα της εντολής.
Μνήμη - μνήμη (3,3)	Μεγάλες εναλλαγές στα μήκη εντολών. Μεγάλες εναλλαγές στο έργο που πραγματοποιείται ανά εντολή. Συνεχής προσβάσεις μνήμης.	

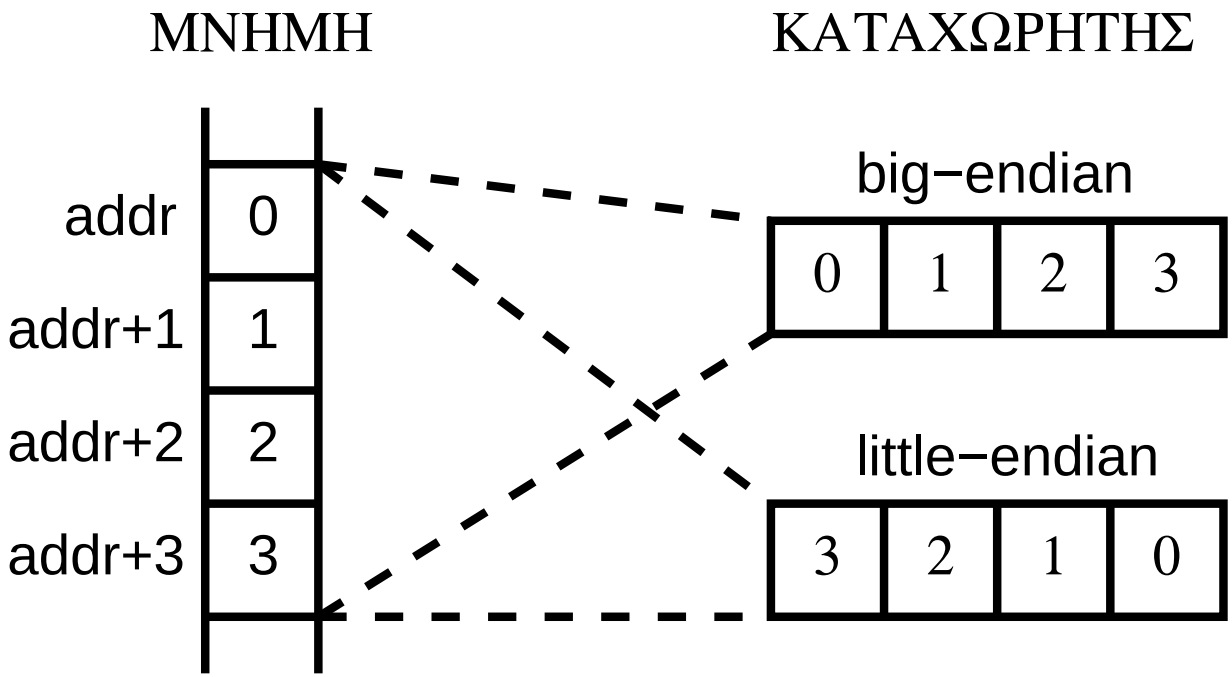
Πρόσβαση Μνήμης

- η πρόσβαση στην μνήμη γίνεται σε μονάδες ***bytes*** (8 ***bits***).
- επίσεις προσφέρονται εντολές για πρόσβαση μισών λέξεων (***half words*** - 16 ***bits***) και λέξεων (***words*** - 32 ***bits***).
- οι πιό πολλές μηχανές επίσεις προσφέρουν πρόσβαση για διπλές λέξεις (***double words*** - 64 ***bits***).

Σύμβασεις για σειρά ***bytes*** σε ***words***:

- μικρό-επί-τέλους, ***little-endian***: το ***byte*** της μικρότερης διεύθυνσης αντιστοιχεί στο τέλος (***LSB***) της λέξης.
- μεγάλο-επί-τέλους, ***big-endian***: το ***byte*** της μεγαλύτερης διεύθυνσης αντιστοιχεί στο τέλος (***LSB***) της λέξης.
- η σειρά των ***bytes*** προκαλεί προβλήματα συμβατότητας οταν μεταφέρονται δεδομένα ανάμεσα σε μηχανές.

Πρόσβαση Μνήμης - Ομαδοποίηση *bytes*



Πρόσβαση Μνήμης - Ευθυγράμμιση

- σε πολλές μηχανές η πρόσβαση σε ποσότητες μεγαλύτερες απο *byte* πρέπει να είναι ευθυγραμισμμένες (*aligned*).

- αυτό συμβαίνει επειδή η μνήμη παρέχει τα δεδομένα σαν *words* ή *doublewords*.

- έτσι, η πρόσβαση σε μή-ευθυγραμμισμένη διεύθυνση θα σημαίνει πάνω απο μία πρόσβαση.

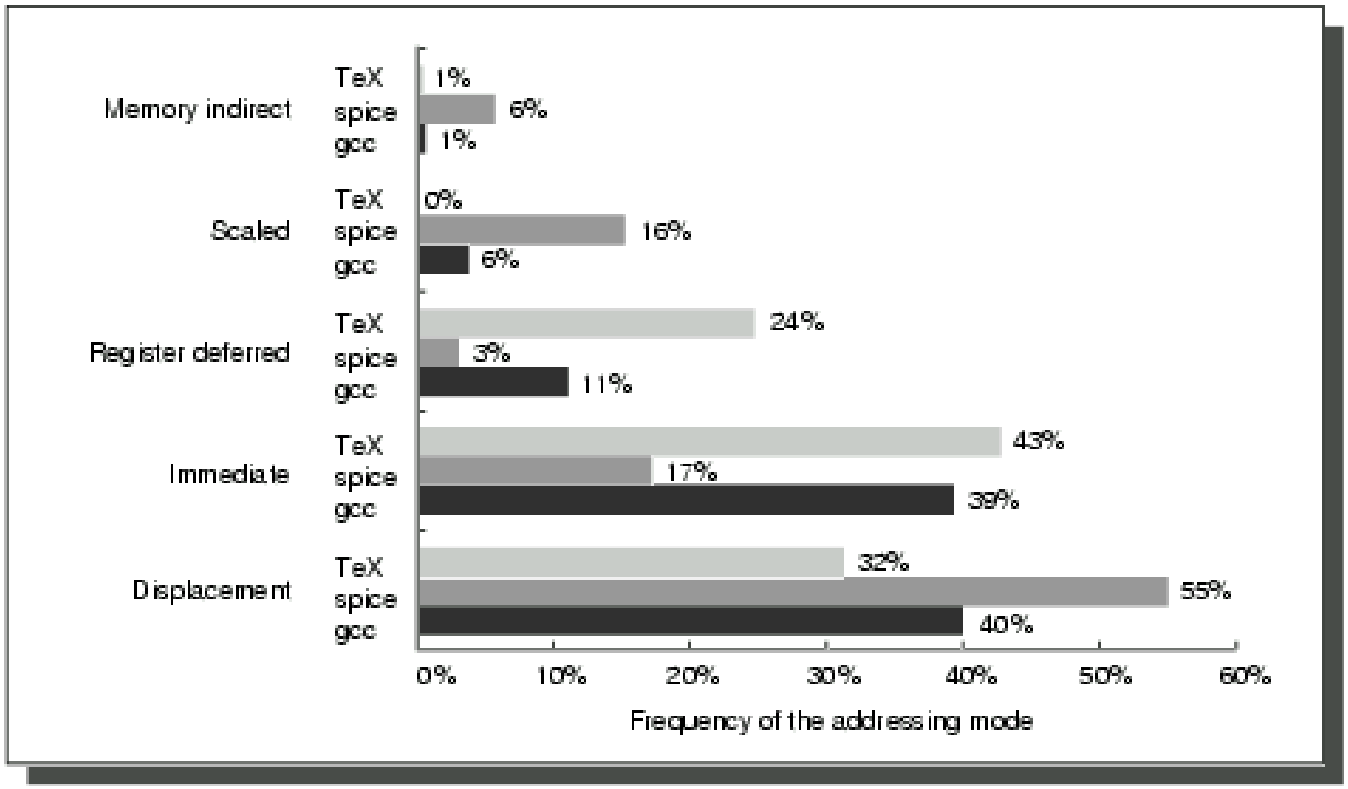
Μέγεθος	Ευθυγραμμισμένο στα προθέματα	Μή-ευθυγραμμισμένο στα προθέματα
Byte	0, 1, 2, 3, 4, 5, 6, 7	Ποτέ
Half Word	0, 2, 4, 6	1, 3, 5, 7
Word	0, 4	1, 2, 3, 5, 6, 7
Double Word	0	1, 2, 3, 4, 5, 6, 7

Τύποι Πρόσβασης Μνήμης - *Memory Addressing*

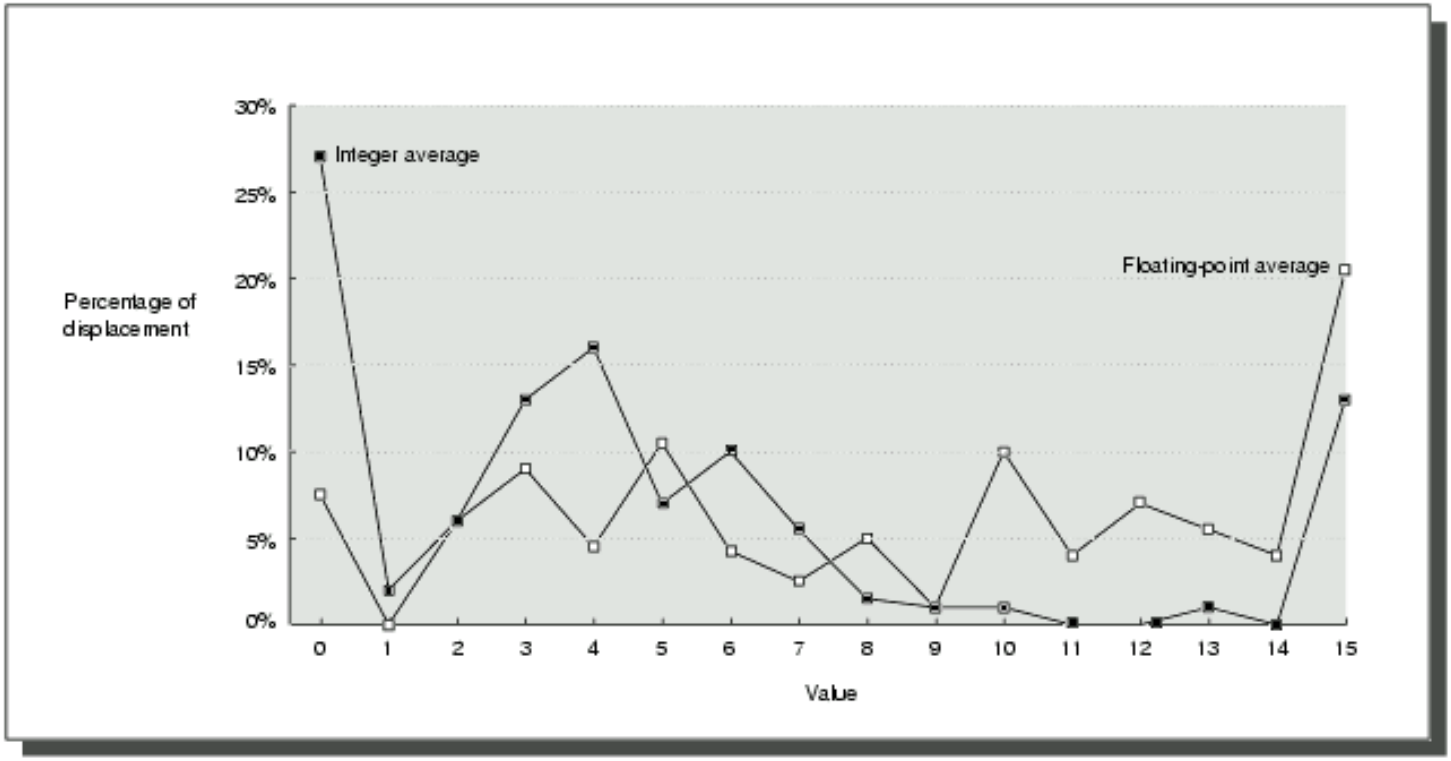
Addressing mode	Example instruction	Meaning	When used
Register	`Add R4,R3`	`Regs[R4]←Regs[R4]+ Regs[R3]`	When a value is in a register.
Immediate	`Add R4,#3`	`Regs[R4]←Regs[R4]+3`	For constants.
Displacement	`Add R4,100(R1)`	`Regs[R4]←Regs[R4]+ Mem[100+Regs[R1]]`	Accessing local variables.
Register deferred or indirect	`Add R4,(R1)`	`Regs[R4]←Regs[R4]+ Mem[Regs[R1]]`	Accessing using a pointer or a computed address.
Indexed	`Add R3,(R1 + R2)`	`Regs[R3]←Regs[R3]+ Mem[Regs[R1]+Regs[R2]]`	Sometimes useful in array addressing: `R1` = base of array; `R2` = index amount.
Direct or absolute	`Add R1,(1001)`	`Regs[R1]←Regs[R1]+ Mem[1001]`	Sometimes useful for accessing static data; address constant may need to be large.
Memory indirect or memory deferred	`Add R1,@(R3)`	`Regs[R1]←Regs[R1]+ Mem[Mem[Regs[R3]]]`	If `R3` is the address of a pointer p, then mode yields $*p$.
Autoincrement	`Add R1,(R2)+`	`Regs[R1]←Regs[R1]+ Mem[Regs[R2]]` `Regs[R2]←Regs[R2]+`d	Useful for stepping through arrays within a loop. `R2` points to start of array; each reference increments `R2` by size of an element, d.
Auto-decrement	`Add R1,-(R2)`	`Regs[R2]←Regs[R2]-`d `Regs[R1]←Regs[R1]+ Mem[Regs[R2]]`	Same use as autoincrement. Autodecrement/increment can also act as push/ pop to implement a stack.
Scaled	`Add R1,100(R2)[R3]`	`Regs[R1]← Regs[R1]+ Mem[100+Regs[R2]+Regs[R3]*`d`]`	Used to index arrays. May be applied to any indexed addressing mode in some machines.

ο VAX11/780 της $Digital$ (1975)

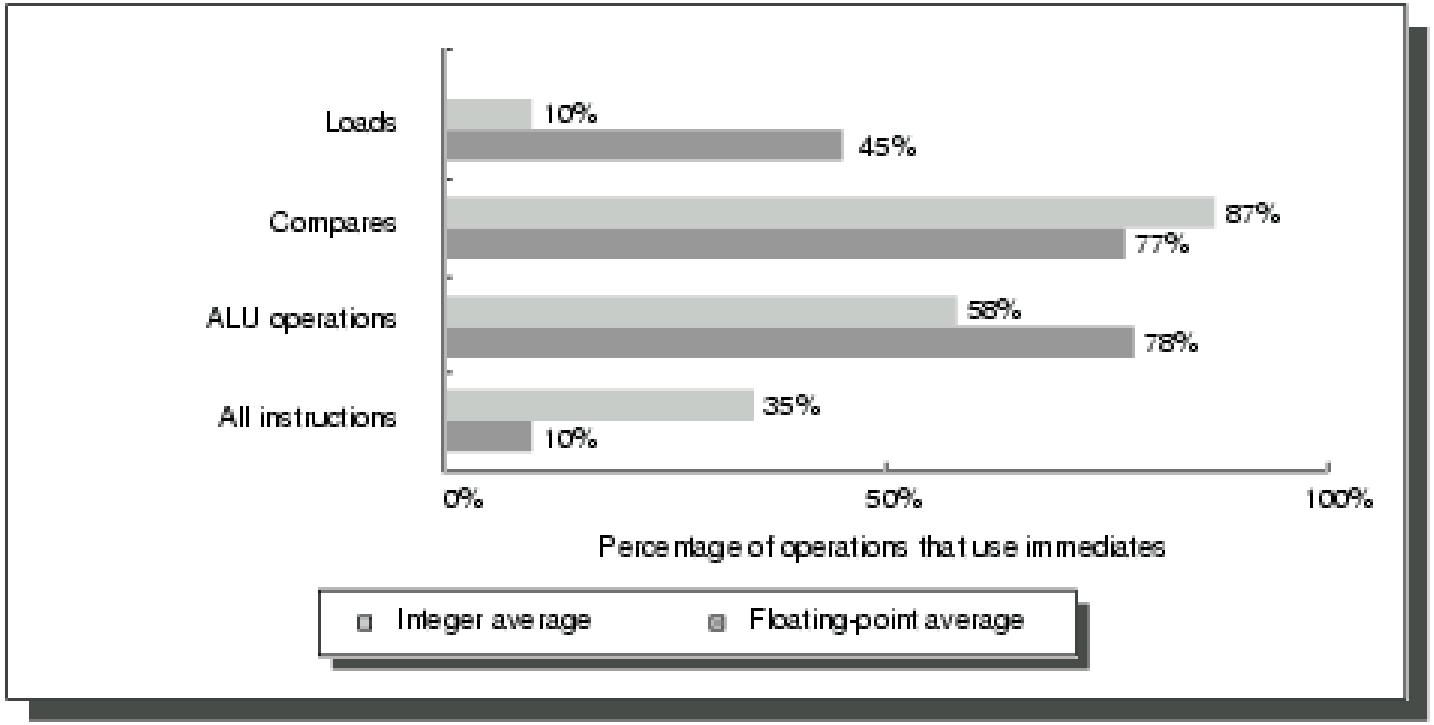
Συχνότητα Τύπων Πρόσβασης Μνήμης στον VAX



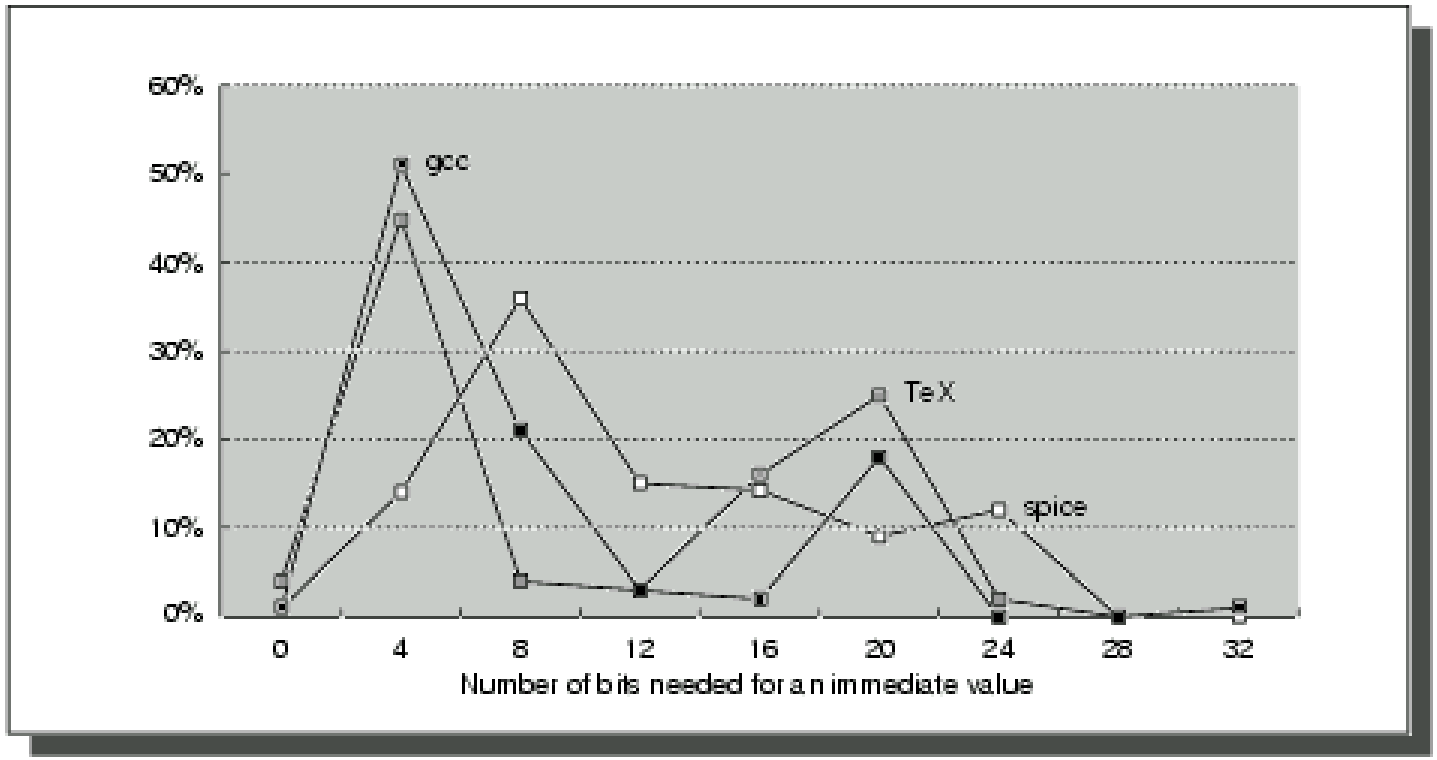
Πρόσβαση με Πρόθεμα - Στατιστική στον $MIPS$



Πρόσβαση με Άμεσο - Στατιστική στον DLX



Μέγεθος Άμεσης Τιμής - Στατιστική στον VAX



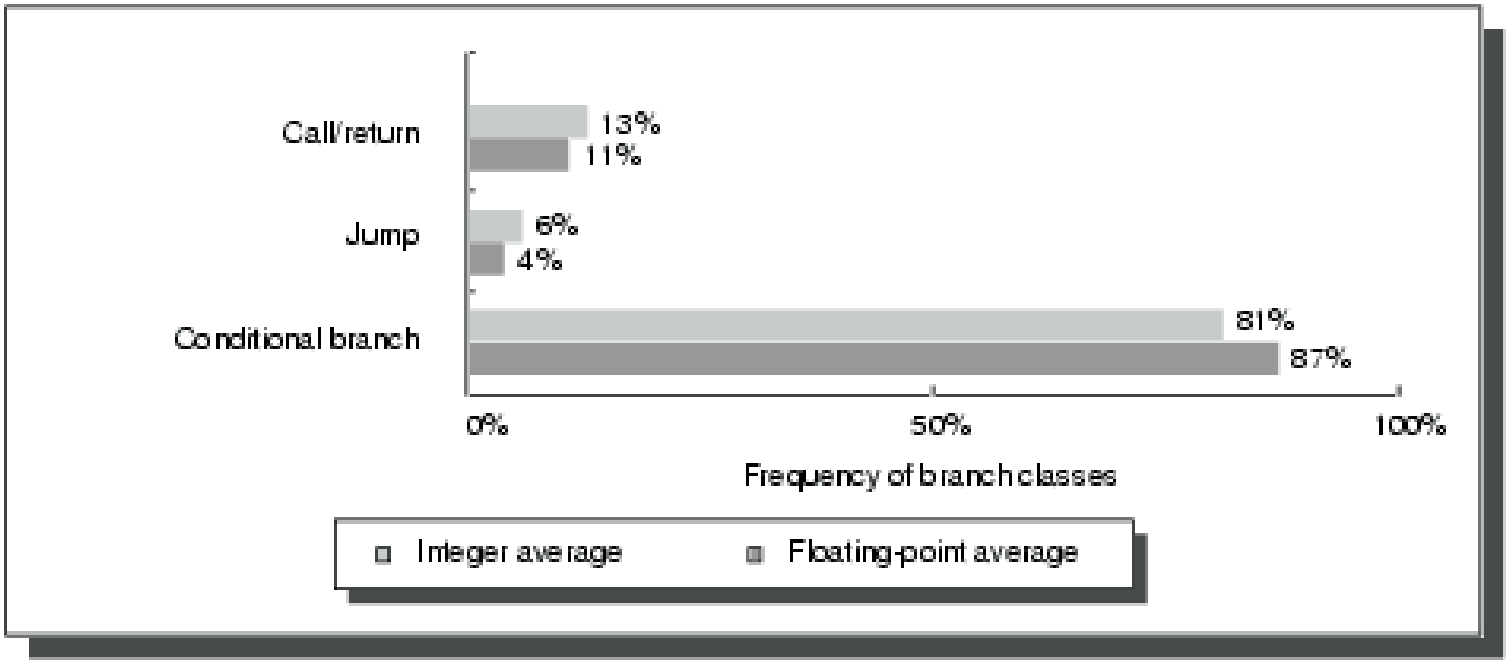
Τύποι Λειτουργιών σε ένα σύνολο εντολών

Τύπος Λειτουργίας	Παραδείγματα
Αριθμητική/Λογική	Αριθμητικές καί λογικές πράξεις: πρόσθεση, σύζευξη, αφαίρεση
Μεταφορά Δεδομένων	Εντολές Φόρτωσης/Αποθήκευσης, εντολές μετακίνησης - *MOV*
Έλεγχος Προγράμματος	Βρόχοι, μεταφορά, κλήσεις και επιστροφή, εξαιρέσεις
Συστήματος	Κλήση λειτουργικού, εντολές χρήσης εικονικής μνήμης
Ρητών	Λειτουργίες Ρητών: πρόσθεση, αφαίρεση
Δεκαδικών	Δεκαδική πρόσθεση, αφαίρεση, μετατροπές
Συμβολοσειρών	Μετακίνηση, σύγκριση, ψάξιμο συμβολοσειρών
Γραφικών	Λειτουργίες ***pixels***, συμπίεση/αποσυμπίεση

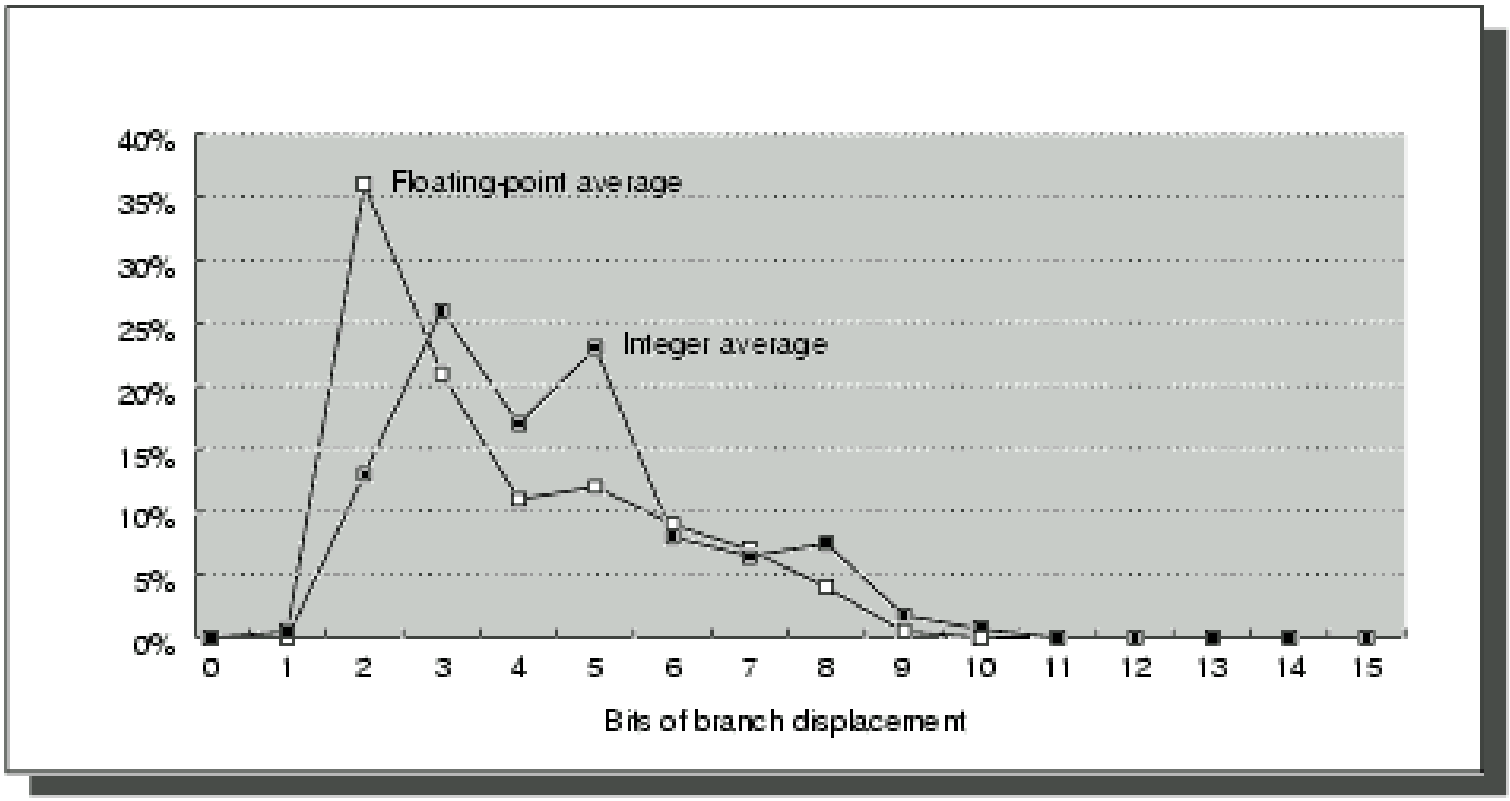
Χρήση των Εντολών στον 80x86

Βαθμός	Εντολή 80x86	Ποσοστό (%)
1	φόρτωσης	22%
2	προυποθετικός βρόχος	20%
3	σύγκριση	16%
4	αποθήκευση	12%
5	πρόσθεση	8%
6	σύζευξη	6%
7	αφαίρεση	5%
8	μεταφορά καταχωρητή-καταχωρητή	4%
9	κλήση	1%
10	επιστροφή	1%

Εντολές Ελέγχου



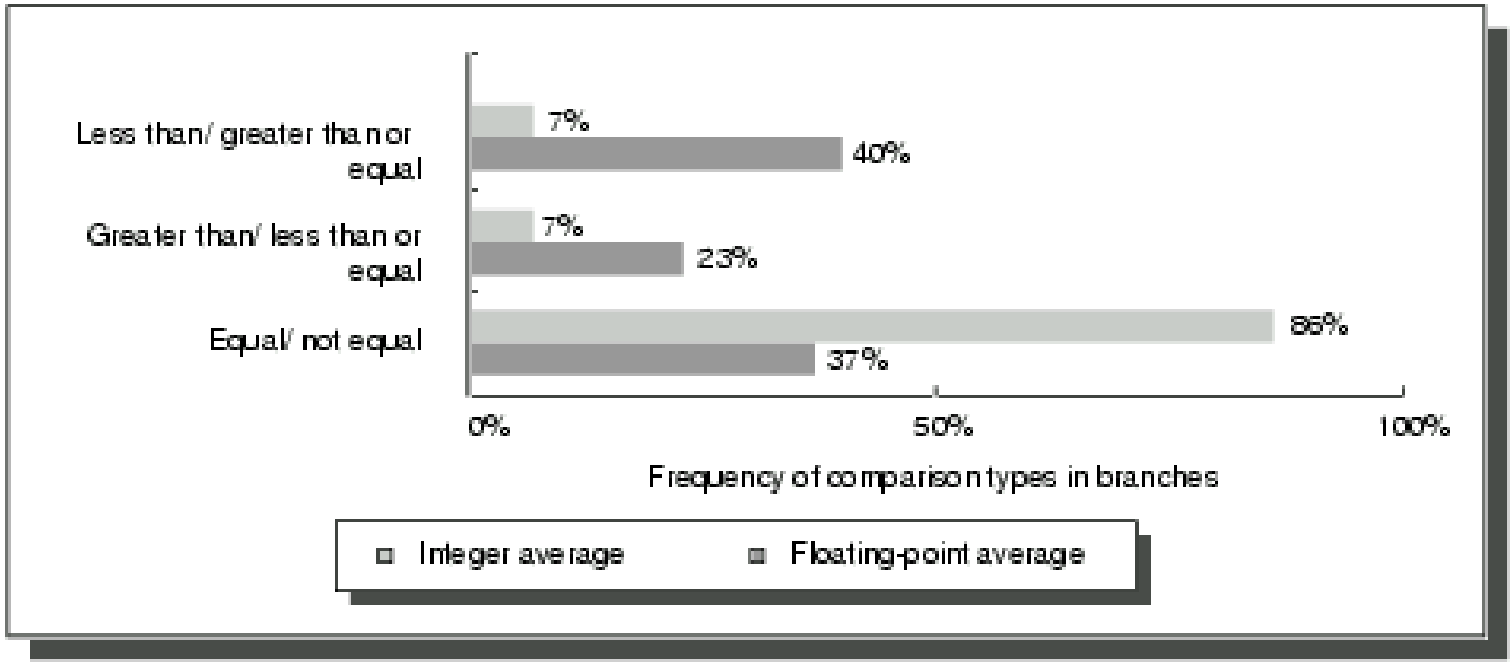
Εντολές Ελέγχου - Αποστάσεις βρόχων σε εντολές (*load-store*)



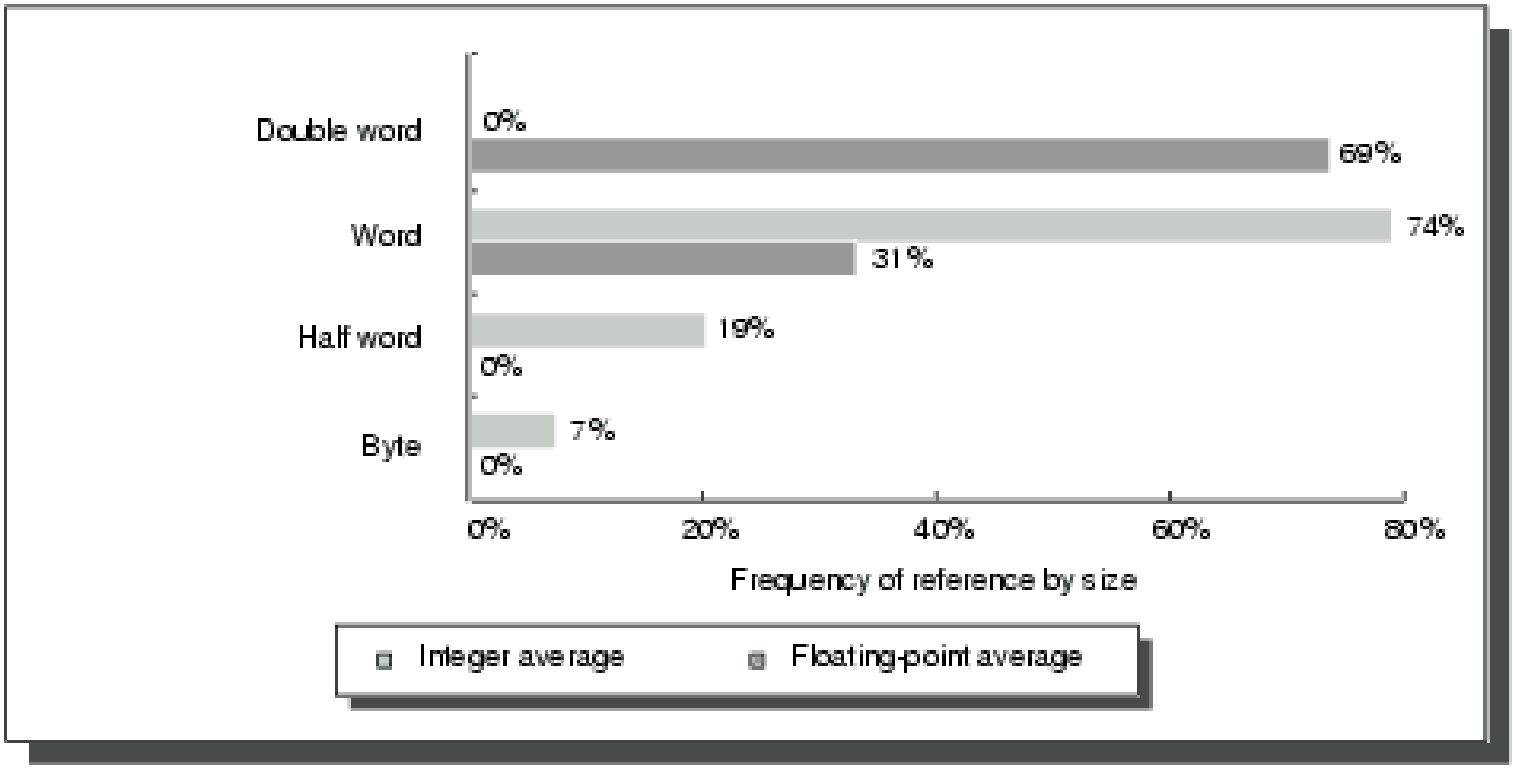
Εντολές Ελέγχου - Μέθοδοι για αξιολόγηση βρόχων

Όνομα	Μέθοδος	Πλεονεκτήματα	Μειονεκτήματα
Condition Code - CC	ειδικά ***bits*** θέτονται απο λειτουργίες ALU.	Δεν χρειάζονται συνεχή αξιολόγηση.	αντιπροσωπεύουν κατάσταση. περιορίζουν την σειρά των εντολών.
Προυποθετικός Καταχωρητής	κάποιος καταχωρητής αποθηκεύει το αποτέλεσμα μιας σύγκρισης.	Απλότητα.	Χρήση Καταχωρητη.
Σύγκριση καί Βρόχος	η σύγκριση είναι μέρους του βρόχου.	Μία εντολή αντί δύο για το βρόχο.	συνεπάγεται αρκετό έργο για την εντολή.

Συχνότητα Συγκρίσεων Βρόχων



Μέγεθος Δρώμενων



Κωδικοποίση ενός συνόλου εντολών

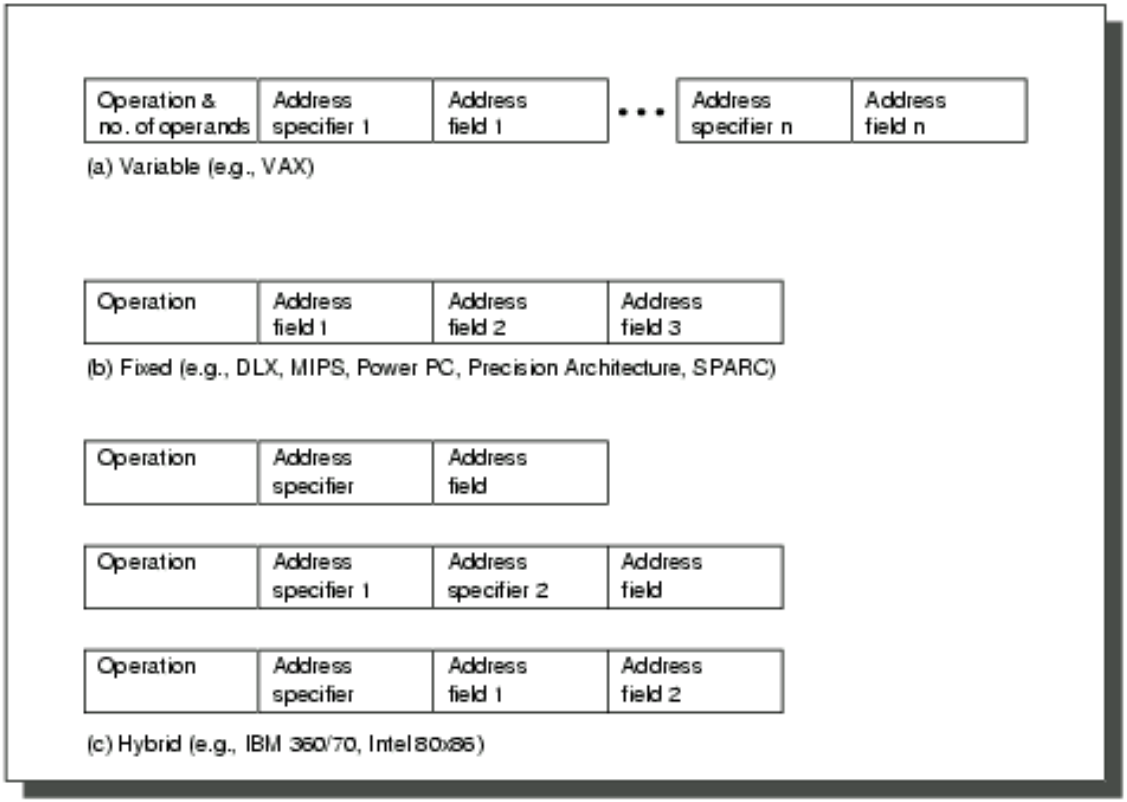
Η κωδικοποίηση των εντολών πρέπει να σταθμίζει τους παρακάτω παράγοντες:

1. Την πρόθεση να έχει όσο το δυνατό πιο πολλούς καταχωρητές και μεθόδους πρόσβασης μνήμης.

2. Το αποτέλεσμα που έχει το μέγεθος των πεδίων των καταχωρητών και μεθόδων πρόσβασης στο μέσο μέγεθος της εντολής και στο μέσο μέγεθος του προγράμματος.

3. Την πρόθεση να είναι εύκολη η απο-κωδικοποίηση των εντολών. Ριζικό ρόλο παίζουν το μήκος μιας εντολής (σταθερό η όχι) και το μέγεθος των πεδίων.

Κωδικοποίση ενός συνόλου εντολών



η Αρχιτεκτονική του DLX

Καταχωρητές:

- ο DLX έχει 32, 32-bit γενικούς καταχωρητές με ονόματα $R0$, $R1$, $R2$, ...
- επίσεις έχει 32, 32-bit ρητούς καταχωρητές με ονόματα $F0$, $F1$, $F2$, ... που μπορούν να χρησιμοποιηθούν και σαν διπλής ακριβείας (64-bit) ώς $F0$, $F2$, ...
- ο καταχωρητής $R0$ είναι βραχυκυκλωμένος στην τιμή μηδέν.

Τύποι Δεδομένων:

- οι τύποι των δεδομένων είναι $bytes$ των 8-$bits$, $half\ words$ των 16-$bits$ και $words$ των 32-$bits$.
- επίσεις ρητοί μονής ακριβείας 32-$bits$ και διπλής ακριβείας 64-$bits$.
- οι εντολές που δουλέυουν σε $bytes$, $half\ words$ φορτώνουν το υπόλοιπο του καταχωρητή με μηδενικά ή επεκτείνουν το bit του πρόσημου.

η Αρχιτεκτονική του *DLX*

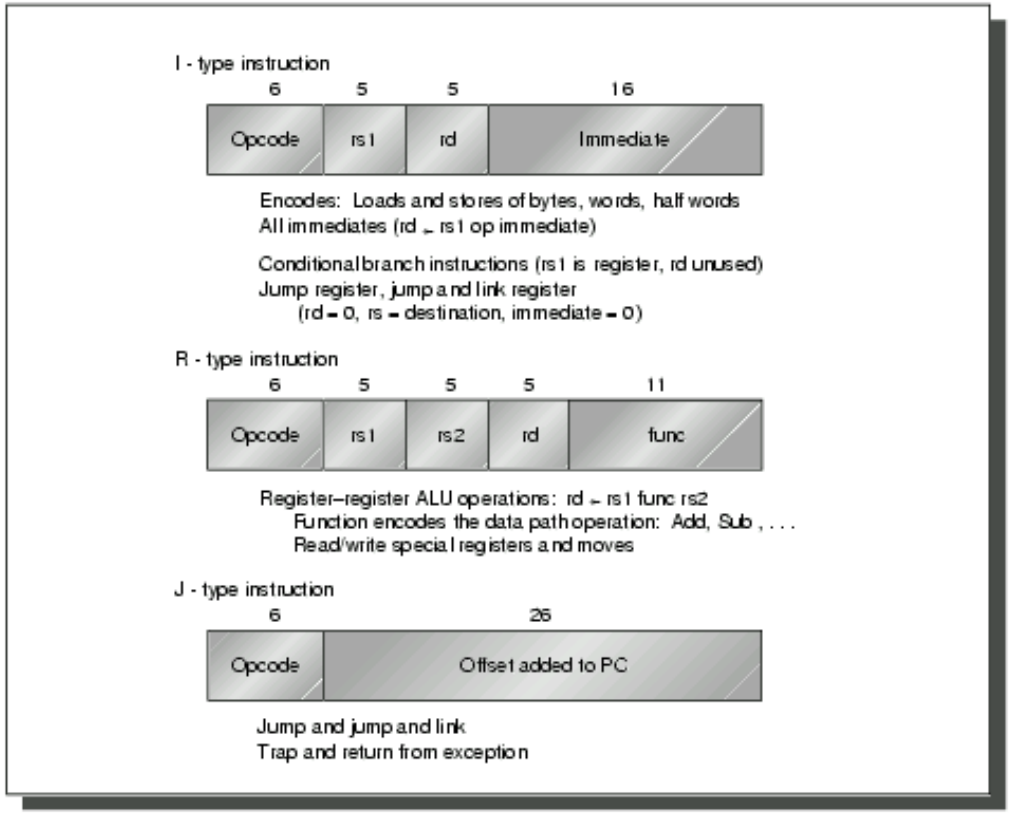
Μέθοδοι Πρόσβασης Μνήμης:

- δύο μόνο μέθοδοι πρόσβασης υποστηρίζονται: με άμεσο (***immediate***) και πρόθεμα (***displacement***).
- το άμεσο και το πρόθεμα έχουν πλάτος 16-***bits***.
- η μνήμη του ***DLX*** ομαδοποιεί τα ***bytes*** με σειρά μεγάλο-επί-τέλους (***big endian***) και χρησιμοποιεί διευθύνσεις 32-***bit***.
- η πρόσβαση γίνεται μόνο με εντολές ***load/store*** και όλες οι διευθύνσεις πρέπει να είναι ευθυγραμμισμένες.

Τύποι Εντολών:

- ο ***DLX*** έχει τέσσερις τύπους εντολών: μνήμης, ***ALU***, βρόχων και ρητών.

Μορφή Εντολών DLX



Εντολές του DLX - $load/store$

Example instruction	Instruction name	Meaning
LW R1,30(R2)	Load word	Regs[R1]$\leftarrow_{32}$ Mem[30+Regs[R2]]
LW R1,1000(R0)	Load word	Regs[R1]$\leftarrow_{32}$ Mem[1000+0]
LB R1,40(R3)	Load byte	Regs[R1]$\leftarrow_{32}$ (Mem[40+Regs[R3]]$_0$)^{24} ## Mem[40+Regs[R3]]
LBU R1,40(R3)	Load byte unsigned	Regs[R1]$\leftarrow_{32}$ 0^{24} ## Mem[40+Regs[R3]]
LH R1,40(R3)	Load half word	Regs[R1]$\leftarrow_{32}$ (Mem[40+Regs[R3]]$_0$)^{16} ## Mem[40+Regs[R3]]##Mem[41+Regs[R3]]
LF F0,50(R3)	Load float	Regs[F0]$\leftarrow_{32}$ Mem[50+Regs[R3]]
LD F0,50(R2)	Load double	Regs[F0]##Regs[F1]$\leftarrow_{64}$ Mem[50+Regs[R2]]
SW 500(R4),R3	Store word	Mem[500+Regs[R4]]$\leftarrow_{32}$ Regs[R3]
SF 40(R3),F0	Store float	Mem[40+Regs[R3]]$\leftarrow_{32}$ Regs[F0]
SD 40(R3),F0	Store double	Mem[40+Regs[R3]]$\leftarrow_{32}$ Regs[F0]; Mem[44+Regs[R3]]$\leftarrow_{32}$ Regs[F1]
SH 502(R2),R3	Store half	Mem[502+Regs[R2]]$\leftarrow_{16}$ Regs[R31]$_{16..31}$
SB 41(R3),R2	Store byte	Mem[41+Regs[R3]]$\leftarrow_{8}$ Regs[R2]$_{24..31}$

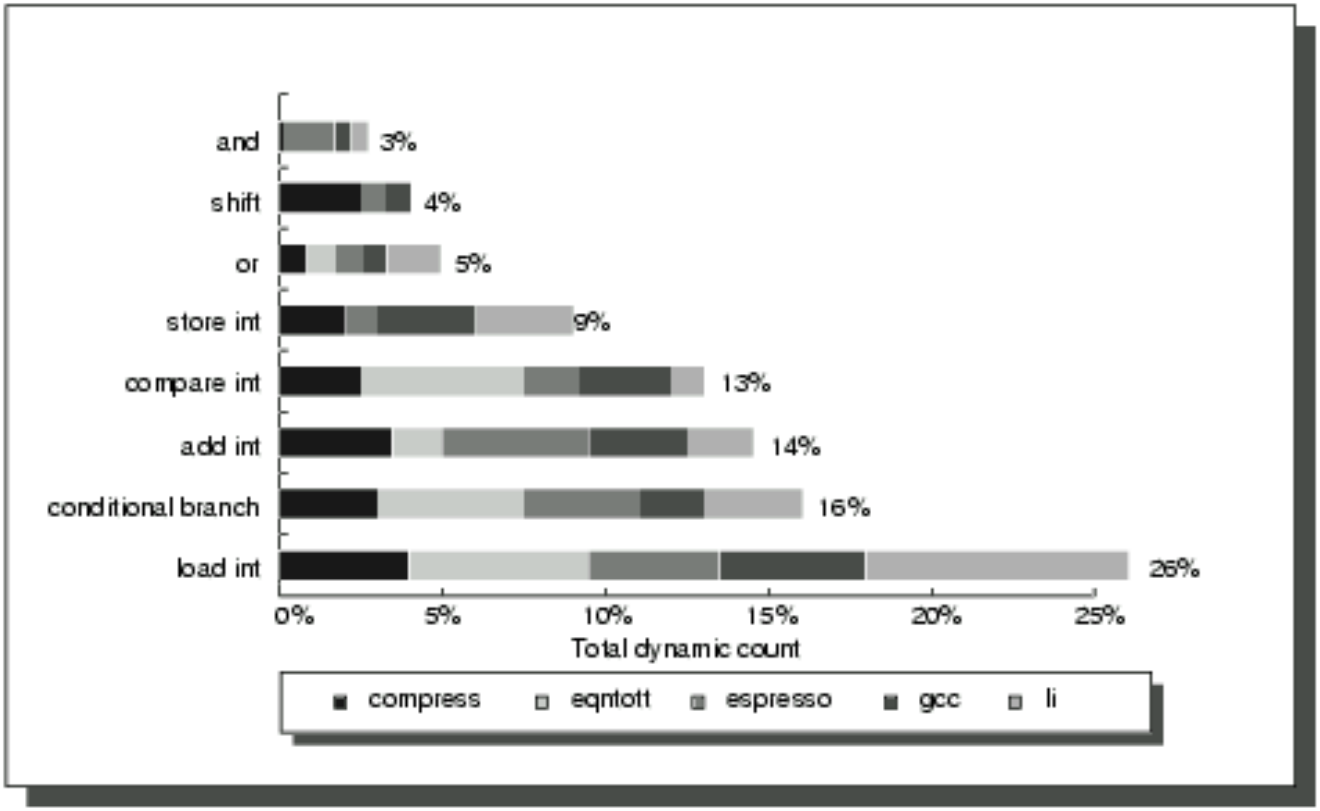
Εντολές του DLX - ALU

Example instruction	Instruction name	Meaning
`ADD R1,R2,R3`	Add	`Regs[R1]←Regs[R2]+Regs[R3]`
`ADDI R1,R2,#3`	Add immediate	`Regs[R1]←Regs[R2]+3`
`LHI R1,#42`	Load high immediate	`Regs[R1]←42##0`^{16}
`SLLI R1,R2,#5`	Shift left logical immediate	`Regs[R1]←Regs[R2]<<5`
`SLT R1,R2,R3`	Set less than	`if (Regs[R2]<Regs[R3]) Regs[R1]←1 else Regs[R1]←0`

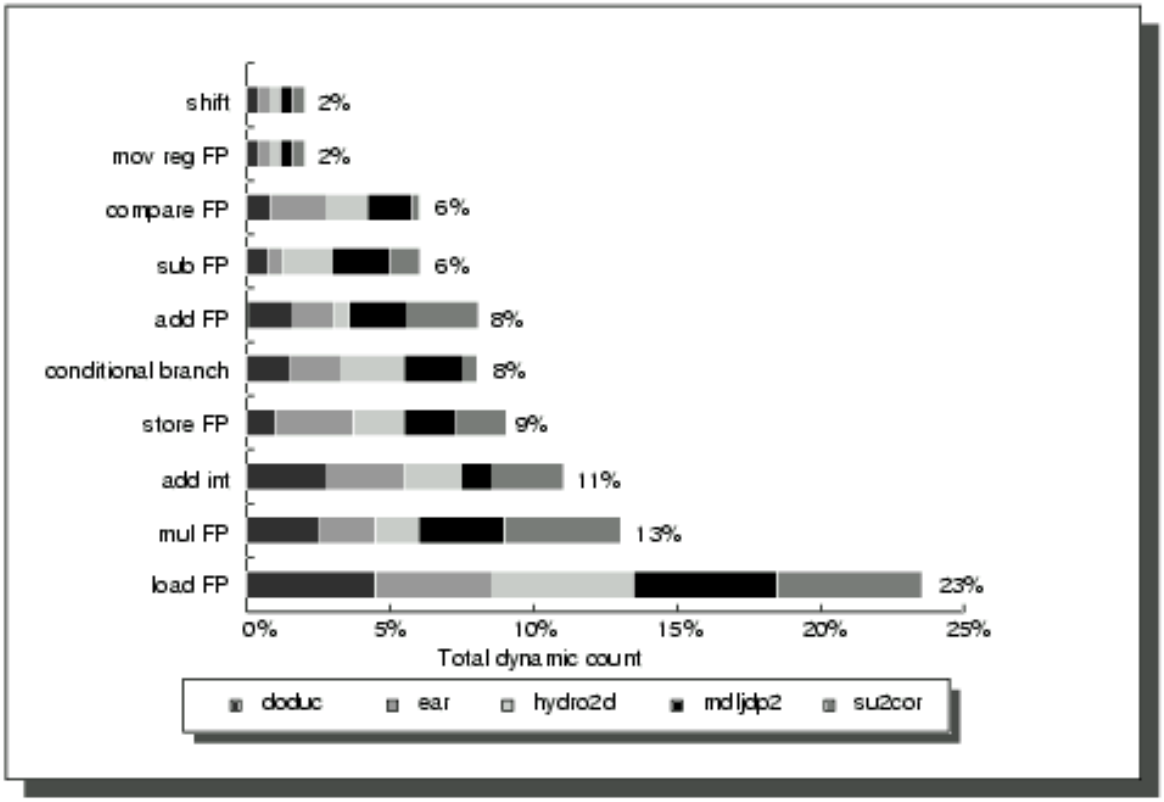
Εντολές του DLX - Ελέγχου

Example instruction	Instruction name	Meaning
J name	Jump	PC←name; ((PC+4)−2^{25}) ≤ name < ((PC+4)+2^{25})
JAL name	Jump and link	R31←PC+4; PC←name; ((PC+4)−2^{25}) ≤ name < ((PC+4)+2^{25})
JALR R2	Jump and link register	Regs[R31]←PC+4; PC←Regs[R2]
JR R3	Jump register	PC←Regs[R3]
BEQZ R4,name	Branch equal zero	if (Regs[R4]==0) PC←name; ((PC+4)−2^{15}) ≤ name < ((PC+4)+2^{15})
BNEZ R4,name	Branch not equal zero	if (Regs[R4]!=0) PC←name; ((PC+4)−2^{15}) ≤ name < ((PC+4)+2^{15})

Στατιστική εντολών DLX για $SPECint92$



Στατιστική εντολών DLX για $SPECfp92$



ο DLX στον προσομοιωτή $HASE$