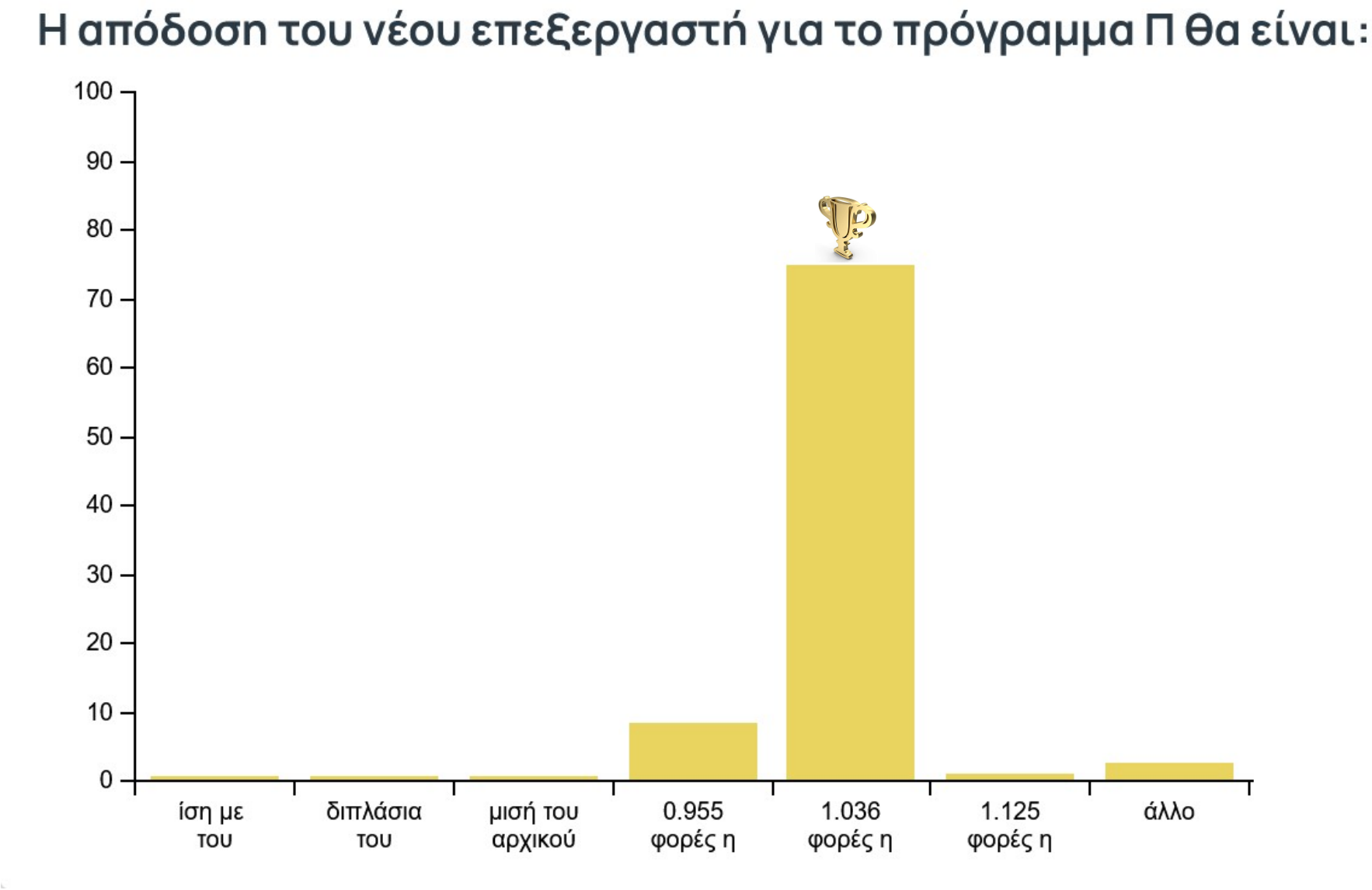
Κουίζ 1 – εκφώνηση

- Σε έναν μικροεπεξεργαστή RISC-V με **χρόνο κύκλου ρολογιού 300 psec** εκτελείται πρόγραμμα Π. Το πρόγραμμα εκτελεί συνολικά **850 τρισεκατομμύρια εντολές** και έχει **CPI ίσο με 2.45** (μέσο αριθμό κύκλων ρολογιού ανά εντολή – clocks per instruction).
- Σχεδιάζεται ένα νέο μοντέλο του μικροεπεξεργαστή (η αρχιτεκτονική συνόλου εντολών είναι η ίδια – RISC-V) το οποίο θα έχει **λίγο μικρότερο χρόνο κύκλου ρολογιού στα 275 psec**. Λόγω αυτού όμως το CPI του προγράμματος Π θα **αυξηθεί σε 2.58** ενώ το πλήθος των εντολών παραμένει το ίδιο εφόσον δεν έχει αλλάξει ο μεταγλωττιστής ή το πηγαίο αρχείο σε γλώσσα υψηλού επιπέδου.
- **Ποια θα είναι η απόδοση του νέου μοντέλου επεξεργαστή όταν εκτελεί το πρόγραμμα Π;**

Λύση

- Ο χρόνος εκτέλεσης του προγράμματος Π στον αρχικό επεξεργαστή είναι:
  - $T = I \times CPI / f_{CLK}$
  - $T = 850 \times 10^{12} \times 2.45 \times 300 \times 10^{-12}$
  - $T = 624\ 750$ s
- Στον νέο επεξεργαστή ο χρόνος εκτέλεσης είναι:
  - $T' = 850 \times 10^{12} \times 2.58 \times 275\ 10^{-12}$
  - $T' = 603\ 075$ s
- Απ' όπου προκύπτει ότι η σχέση της απόδοσης του προγράμματος Π στους δυο επεξεργαστές είναι:
- **$Απόδοση_{ΝΕΟΥ}/Απόδοση_{ΑΡΧΙΚΟΥ} = Χρόνος_{ΑΡΧΙΚΟΥ} / Χρόνος_{ΝΕΟΥ} = 624\ 750 / 603\ 075 = 1.0359$**
- **Η απόδοση του νέου είναι 1.0359 φορές αυτή του αρχικού**

Απαντήσεις (299 συμμετοχές)



Κουίζ 2 – εκφώνηση

- Τρεις φοιτητές πληροφορικής Φ1, Φ2, Φ3 αγόρασαν τους φορητούς υπολογιστές C1, C2, C3 πληρώνοντας 1000, 1100, 1200 ευρώ, αντίστοιχα. Οι επεξεργαστές έχουν ρυθμό ρολογιού 2.5 GHz, 2.6 GHz, και 2.8 GHz, αντίστοιχα. Οι τρεις επεξεργαστές είναι της ίδιας αρχιτεκτονικής συνόλου εντολών αλλά διαφορετικής μικροαρχιτεκτονικής και όταν εκτελούν το ίδιο πρόγραμμα Π, αυτό έχει CPI 2.9 στον πρώτο, 3.0 στον δεύτερο, και 3.1 στον τρίτο. Το πρόγραμμα Π εκτελεί και στους τρεις επεξεργαστές 750 τρισεκατομμύρια εντολές.
- **Ποιος από τους τρεις φοιτητές παίρνει από τον υπολογιστή του τον καλύτερο (μεγαλύτερο) λόγο «Απόδοση προς Κόστος» για την εκτέλεση του προγράμματος Π;**

Λύση

- Ο χρόνος εκτέλεσης του Π στον C1 είναι
- $\text{Χρόνος}_{C1} = I \times \text{CPI}_{C1} / \text{ClockRate}_{C1} = (750 \times 10^{12} \times 2.9) / (2.5 \times 10^{9})$ = 870 000 sec
- Ομοίως των C2 και C3
- $\text{Χρόνος}_{C2} = I \times \text{CPI}_{C2} / \text{ClockRate}_{C2} = (750 \times 10^{12} \times 3.0) / (2.6 \times 10^{9})$ = 865 385 sec
- $\text{Χρόνος}_{C3} = I \times \text{CPI}_{C3} / \text{ClockRate}_{C3} = (750 \times 10^{12} \times 3.1) / (2.8 \times 10^{9})$ = 830 357 sec
- Ο λόγος Απόδοσης προς Κόστος για τους τρεις υπολογιστές είναι
- $\Lambda_{C1} = \text{Απόδοση}_{C1}/\text{Κόστος}_{C1} = 1/(\text{Χρόνος}_{C1} \times \text{Κόστος}_{C1}) =$ **1.15×10^{-9}**
- $\Lambda_{C2} = \text{Απόδοση}_{C2}/\text{Κόστος}_{C2} = 1/(\text{Χρόνος}_{C2} \times \text{Κόστος}_{C2}) =$ **1.05×10^{-9}**
- $\Lambda_{C3} = \text{Απόδοση}_{C3}/\text{Κόστος}_{C3} = 1/(\text{Χρόνος}_{C3} \times \text{Κόστος}_{C3}) =$ **1.00×10^{-9}**
- **Νικητής ο φοιτητής Φ1 ο οποίος παίρνει υψηλότερη απόδοση για το πρόγραμμα Π ανά μονάδα κόστους που πλήρωσε.**

Απαντήσεις (290 συμμετοχές)

Κουίζ 3 – εκφώνηση

- Μια CPU R1 αρχιτεκτονικής συνόλου εντολών RISC-V εκτελεί ένα πρόγραμμα που αποτελείται από 15% εντολές τύπου Α με CPI=1, άλλο 35% εντολές τύπου Β με CPI=2, άλλο ένα 40% εντολές τύπου Γ με CPI=3, και τέλος 10% εντολές τύπου Δ με CPI=2. Οι συνολικές δυναμικές εντολές του προγράμματος είναι 25 δισεκατομμύρια και ο ρυθμός ρολογιού της R1 είναι 2.5GHz. Μια άλλη CPU R2 με διαφορετική μικροαρχιτεκτονική εκτελεί το ίδιο πρόγραμμα μεταγλωττισμένο από διαφορετικό μεταγλωττιστή. Οι δυναμικές εντολές είναι τώρα 23.5 δισεκατομμύρια και οι συχνότητες τους και τα CPI τους γίνονται τώρα: τύπος Α (10% με CPI=2), τύπος Β (20% με CPI=2), τύπος Α (40% με CPI=4), τύπος Α (30% με CPI=1) ενώ ο ρυθμός ρολογιού της R2 είναι 2.75 GHz.
- **Ποια από τις δύο CPU εκτελεί μεγαλύτερο αριθμό δυναμικών εντολών στην μονάδα του χρόνου (millions of instructions per second);**
- **Ποια από τις δύο CPU εκτελεί το πρόγραμμα ταχύτερα;**

Λύση

- Θα βρούμε τον χρόνο εκτέλεσης του προγράμματος σε καθεμία από τις δύο CPU R1 και R2.

Χρόνος$_{R1}$=Εντολές$_{R1}$ x CPI$_{R1}$ / ΡυθμόςΡολογιού$_{R1}$ =

= 25 x 10^9 x (0.15x1 + 0.35x2 + 0.40x3 + 0.10x2) / (2.5 x 10^9)

= **22.50 sec**

Χρόνος$_{R2}$=Εντολές$_{R2}$ x CPI$_{R2}$ / ΡυθμόςΡολογιού$_{R2}$ =

= 23.5 x 10^9 x (0.10x2 + 0.20x2 + 0.40x4 + 0.30x1) / (2.75 x 10^9)

= **21.36 sec**

- Το πλήθος δυναμικών εντολών που εκτελεί κάθε CPU ανά δευτερόλεπτο (μετρημένο σε MIPS=millions of instructions per second) προκύπτει προφανώς διαιρώντας το πλήθος εντολών με τον χρόνο εκτέλεσης

MIPS$_{R1}$ = 25x10^9/22.50 = 1 111 MIPS

MIPS$_{R2}$ = 23.5x10^9/21.36 = 1 000 MIPS

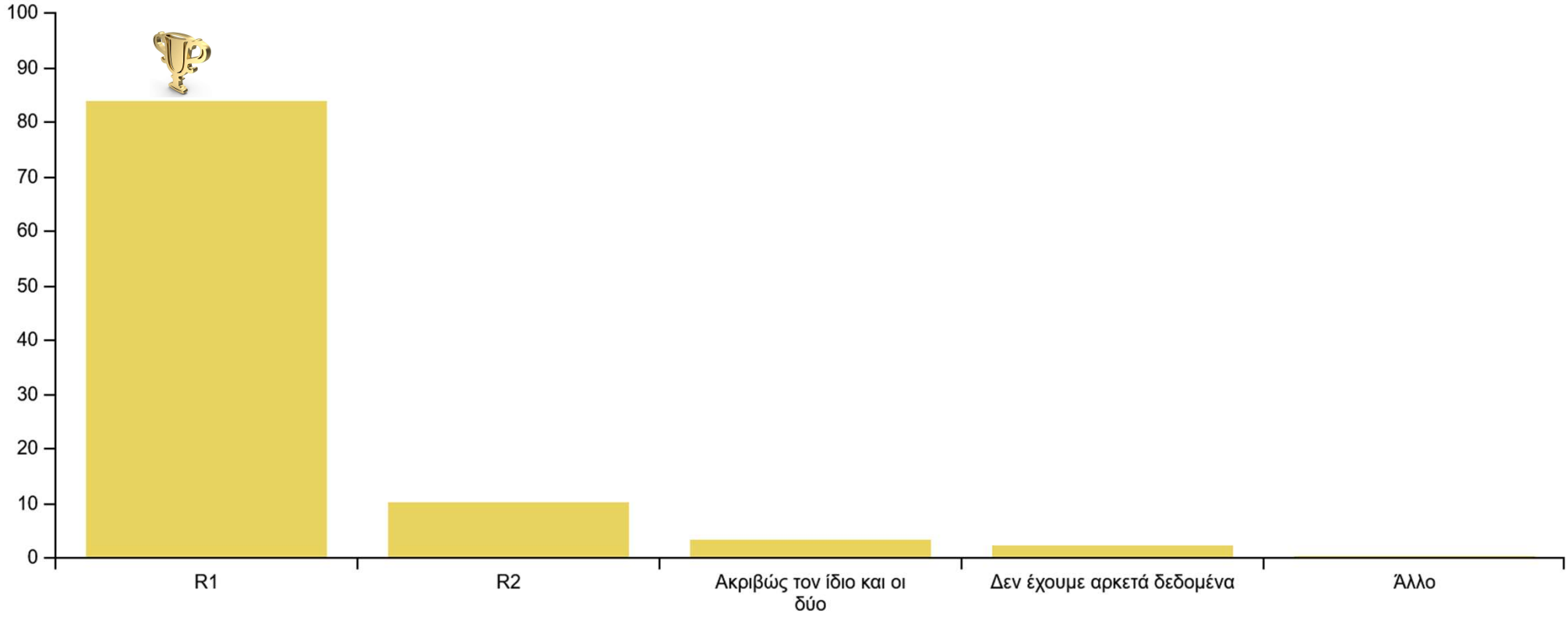
- **(συνεχίζεται ...)**

Λύση (συνέχεια)

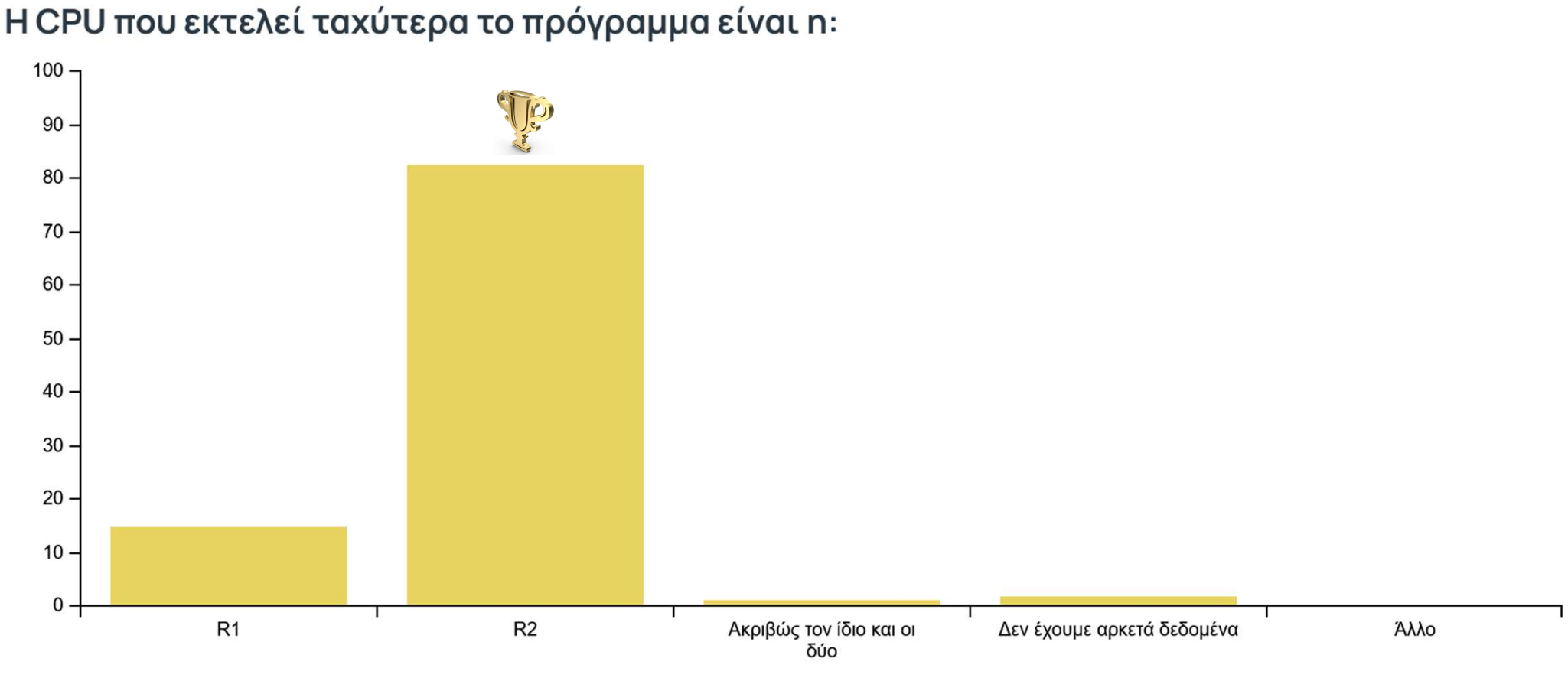
- Συνεπώς:
- **Η CPU R1 εκτελεί περισσότερες εντολές ανά δευτερόλεπτο από την CPU R2 (άρα «δείχνει» να είναι ταχύτερη), αλλά**
- **Η CPU R2 είναι φυσικά η ταχύτερη αφού εκτελεί το πρόγραμμα σε μικρότερο χρόνο**
- Συμπέρασμα: το μόνο σωστό μέτρο σύγκριση της απόδοσης δύο επεξεργαστών είναι ο χρόνος εκτέλεσης. Η μέτρηση MIPS είναι ένα χρήσιμο διαισθητικά εργαλείο (υψηλότερη τιμή MIPS δίνει την αίσθηση ενός ταχύτερου επεξεργαστή) αλλά μπορεί να δώσει παραπλανητικά αποτελέσματα εάν χρησιμοποιηθεί για να συγκριθούν επεξεργαστές με διαφορετικό ρυθμό ρολογιού και διαφορετικά CPI των κατηγοριών εντολών (δηλαδή διαφορετικής μικροαρχιτεκτονικής).

Απαντήσεις (273 συμμετοχές)

Τον μεγαλύτερο αριθμό δυναμικών εντολών στην μονάδα του χρόνου (millions of instructions per second) εκτελεί η CPU;



Απαντήσεις (273 συμμετοχές)



Κουίζ 4 – εκφώνηση

- Μέχρι σήμερα έχουμε διδαχθεί τις εξής εντολές συμβολικής γλώσσας RISC-V: add, sub, addi, lw, sw, beq, bne, jal, jalr, and, andi, or, ori, xor, xori, slli, srli. Μπορείτε να χρησιμοποιήσετε αυτές τις εντολές και μόνο τους επτά καταχωρητές x0 και x5 … x10.
- Γράψτε πρόγραμμα συμβολικής γλώσσας με τις λιγότερες εντολές που μπορείτε που να «περιστρέφει» τα περιεχόμενα των καταχωρητών x5 έως x10. Δηλαδή στο τέλος του προγράμματος ο x5 να περιέχει την αρχική τιμή του x6, ο x6 να περιέχει την αρχική τιμή του x7, …, ο x9 να περιέχει την αρχική τιμή του x10, και ο x10 να περιέχει την αρχική τιμή του x5.
- **Ποιος είναι ο αριθμός των εντολών του προγράμματός σας;**

Λύση

- Με βάση την εκφώνηση δεν υπάρχουν διαθέσιμοι καταχωρητές για να υλοποιηθεί λειτουργία αντιμετάθεσης (swap). Ο x0 δεν μπορεί να χρησιμοποιηθεί για αυτόν τον σκοπό διότι είναι σταθερός στην τιμή 0. Ομοίως, δεν μπορεί να χρησιμοποιηθεί κάποια περιοχή μνήμης (π.χ. η στοίβα) για την αντιμετάθεση τιμών) επειδή ούτε ο καταχωρητής sp (x2) είναι διαθέσιμος.
- Μόνη λύση είναι η αντιμετάθεση των περιεχομένων «διπλανών» καταχωρητών με χρήση εντολών xor.
- **Ο μικρότερος αριθμός εντολών για να επιτευχθεί το ζητούμενο είναι 15. Ο κώδικας στην επόμενη σελίδα.**

Λύση (κώδικας)

- Κάθε τριάδα εντολών με το ίδιο χρώμα, αντιμεταθέτει τα περιεχόμενα των καταχωρητών που μετέχουν στις εντολές.
- **Στο τέλος της εκτέλεσης ο x5 περιέχει την αρχική τιμή του x6, ο x6 περιέχει την αρχική τιμή του x7, ο x7 περιέχει την αρχική τιμή του x8, ο x8 περιέχει την αρχική τιμή του x9, ο x9 περιέχει την αρχική τιμή του x10, και ο x10 περιέχει την αρχική τιμή του x5.**

```
xor x5, x5, x10
xor x10, x10, x5
xor x5, x5, x10
xor x5, x5, x9
xor x9, x9, x5
xor x5, x5, x9
xor x5, x5, x8
xor x8, x8, x5
xor x5, x5, x8
xor x5, x5, x7
xor x7, x7, x5
xor x5, x5, x7
xor x5, x5, x6
xor x6, x6, x5
xor x5, x5, x6
```

Απαντήσεις (262 συμμετοχές)

Κουίζ 5 – εκφώνηση

- Ένας πίνακας με όνομα M αποθηκεύει σε διαδοχικά byte στη μνήμη ενός επεξεργαστή RISC-V 20 απρόσημους αριθμούς M[0], …, M[19] καθένας από τους οποίους έχει τιμή μεταξύ 0 και 12. Θεωρήστε ότι ήδη έχετε στον καταχωρητή x9 τη διεύθυνση μνήμης στην οποία είναι αποθηκευμένος ο πίνακας M (αυτό επιτυγχάνεται με την ψευδοεντολή la).
- Γράψτε πρόγραμμα συμβολικής γλώσσας RISC-V με τις λιγότερες δυνατές εντολές που να υπολογίζει το άθροισμα των αριθμών M[0] έως M[19] και να το αποθηκεύει στον καταχωρητή x10. Μπορείτε να χρησιμοποιήσετε όποιους καταχωρητές και εντολές θέλετε εκτός από εντολές διακλάδωσης και άλματος (branch, jump). Στο πλήθος εντολών μην υπολογίζετε την la και τις εντολές τερματισμού του προγράμματος.
- **Ποιος είναι ο αριθμός των εντολών του προγράμματός σας;**

Λύση (1)

- Η ιδέα της λύσης είναι η εκμετάλλευση του γεγονότος ότι οι αριθμοί είναι πολύ μικροί (τιμές 0 έως 12). Εξαιτίας αυτού μπορούν να προστίθενται «παράλληλα» ως ένα «διάνυσμα» μικρών αριθμών (ένα byte ο καθένας). Για να γίνει αυτό πρέπει τα διαδοχικά byte να φορτωθούν από τη μνήμη ως λέξεις (4άδες byte).
- Ο κώδικας υποθέτει (με βάση την εκφώνηση) ότι στον x9 βρίσκεται ήδη η διεύθυνση του πίνακα Μ (αυτό επιτυγχάνεται με μια ψευδοεντολή la).
- **Συνολικά 17 εντολές**

```
lw x4,0(x9)
lw x5,4(x9)
lw x6,8(x9)
lw x7,12(x9)
lw x8,16(x9)
add x4,x4,x5
add x6,x6,x7
add x4,x4,x6
add x8,x4,x8
srli x4,x8,16
slli x5,x8,16
srli x5,x5,16
add x8,x4,x5
srli x4,x8,8
slli x5,x8,24
srli x5,x5,24
add x10,x4,x5
```

Λύση (2) - δοκιμάστε στον RARS!

```
lw x4,0(x9)
lw x5,4(x9)
lw x6,8(x9)
lw x7,12(x9)
lw x8,16(x9)
add x4,x4,x5
add x6,x6,x7
add x4,x4,x6
add x8,x4,x8
srli x4,x8,16
slli x5,x8,16
srli x5,x5,16
add x8,x4,x5
srli x4,x8,8
slli x5,x8,24
srli x5,x5,24
add x10,x4,x5
```

Ως εδώ οι x4, x5, x6, x7, x8 έχουν λάβει τους 20 αριθμούς (τα 20 byte), 4 αριθμούς κάθε καταχωρητής

Στο σημείο αυτό ο x8 περιέχει στα 4 byte του το άθροισμα των τεσσάρων 5άδων αριθμών

Στο σημείο αυτό ο x8 περιέχει στα 2 δεξιότερα byte του το άθροισμα των δύο 10άδων αριθμών

Στο σημείο αυτό ο x10 περιέχει το άθροισμα των 20 αριθμών

Λύση (3) –
καλύτερη με 14 εντολές (δική σας!)

```
lw x4,0(x9)
lw x5,4(x9)
lw x6,8(x9)
lw x7,12(x9)
lw x8,16(x9)
add x4,x4,x5
add x6,x6,x7
add x4,x4,x6
add x8,x4,x8
srli x4,x8,16
add x8,x4,x8
srli x4,x8,8
add x10,x4,x8
andi x10,x10,0xff
```

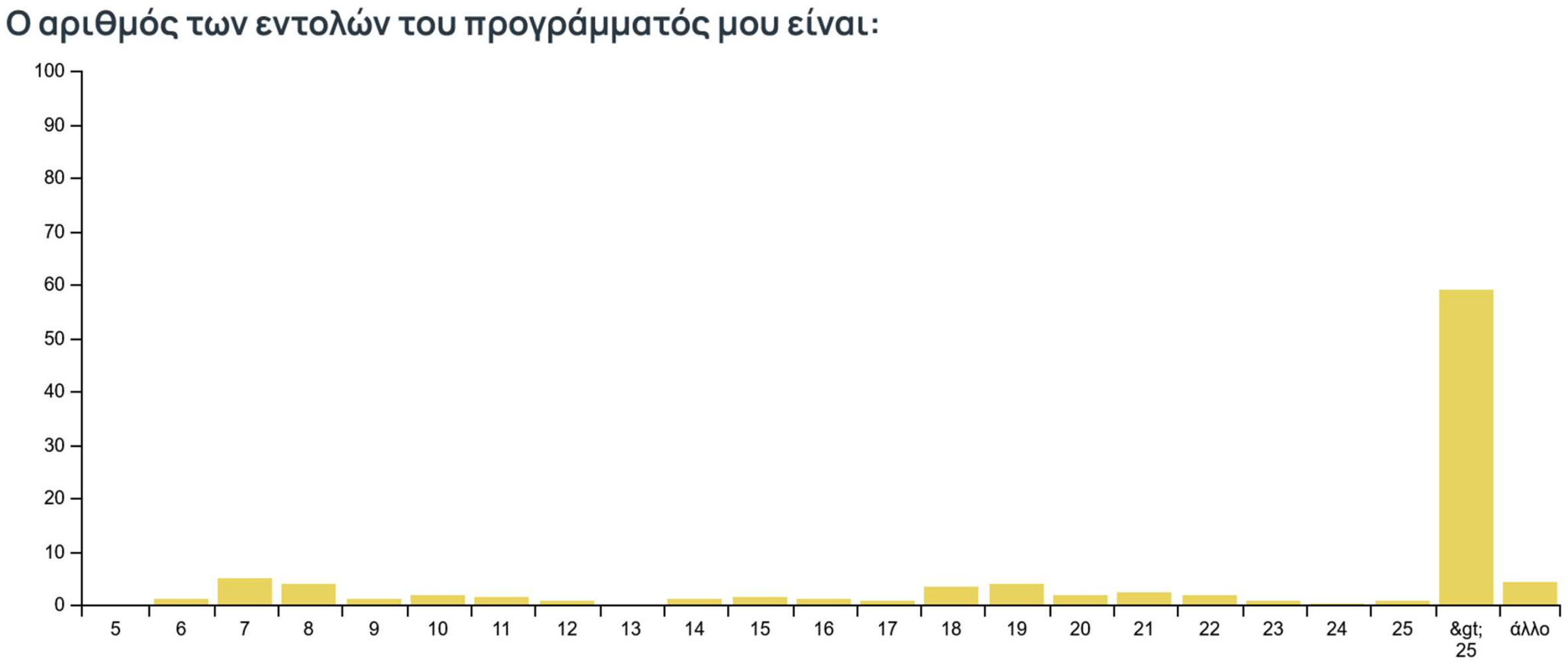
Ως εδώ οι x4, x5, x6, x7, x8 έχουν λάβει τους 20 αριθμούς (τα 20 byte), 4 αριθμούς κάθε καταχωρητής

Στο σημείο αυτό ο x8 περιέχει στα 4 byte του το άθροισμα των τεσσάρων 5άδων αριθμών

Στο σημείο αυτό ο x8 περιέχει στα 2 δεξιότερα byte του το άθροισμα των δύο 10άδων αριθμών

Στο σημείο αυτό ο x10 περιέχει το άθροισμα των 20 αριθμών

Απαντήσεις (252 συμμετοχές)



Ανεβάσατε πολλές ενδιαφέρουσες λύσεις.
Συνεχίστε έτσι.

Κουίζ 6 – εκφώνηση

- Οι μεταβλητές a, b, c, d ενός προγράμματος που θα εκτελεστεί σε επεξεργαστή RISC-V βρίσκονται ήδη στους καταχωρητές x6, x7, x8, x9 (δηλαδή έχουν ήδη εκτελεστεί οι εντολές la και lw που φέρνουν τις τιμές τους από τη μνήμη στους καταχωρητές). Οι αριθμοί είναι απρόσημοι.
- Να γράψετε το υπόλοιπο του προγράμματος που υπολογίζει την τιμή της παράστασης a*b + c*d χωρίς να χρησιμοποιήσετε εντολές πολλαπλασιασμού ή διαίρεσης (μπορείτε να υποθέσετε ότι δεν υπάρχει ενδεχόμενο υπερχείλισης οπότε δεν χρειάζεται να κάνετε κανέναν σχετικό έλεγχο). Μπορείτε να χρησιμοποιήσετε όλες τις υπόλοιπες εντολές που μάθαμε ως τώρα και όποιους άλλους καταχωρητές θέλετε. Η τελική τιμή της παράστασης να αποθηκεύεται στον καταχωρητή x10. Το πρόγραμμά σας να αποτελείται από τις λιγότερες δυνατές εντολές. Στο πλήθος εντολών μην υπολογίζετε τις εντολές τερματισμού του προγράμματος ούτε φυσικά τις αρχικές la και lw.
- **Ποιος είναι ο αριθμός των εντολών του προγράμματός σας;**

Λύση (1)

- Ο κώδικας περιλαμβάνει δύο βρόχους. Ο πρώτος υπολογίζει το axb με διαδοχικές αφαιρέσεις από το b και προσθέσεις του a στο γινόμενο. Ομοίως ο δεύτερος υπολογίζει το cxd. Στο τέλος τα δύο γινόμενα προστίθενται.
- **Συνολικά 9 εντολές (υπολογίζονται μόνο οι χρωματισμένες με βάση τις οδηγίες της εκφώνησης).**
- **Σε ποιες περιπτώσεις δεν δουλεύει σωστά το πρόγραμμα;**

Λύση (2)

```
        .text
        .globl __start
__start:
        la x3, a
        lw x6, 0(x3)
        la x3, b
        lw x7, 0(x3)
        la x3, c
        lw x8, 0(x3)
        la x3, d
        lw x9, 0(x3)
        add x11, x0, x0             # clear the product
axb:    add x11, x11, x6    # add multiplicant to product
        addi x7, x7, -1             # minus 1 the multiplier
        bgt x7, x0, axb             # continue if > zero
        add x12, x0, x0             # clear the product
cxd:    add x12, x12, x8    # add multiplicant to product
        addi x9, x9, -1             # minus 1 the multiplier
        bgt x9, x0, cxd             # continue if > zero
        add x10, x11, x12
        li  a7, 10                  # Done, terminate program
        ecall                       # au revoir...
```

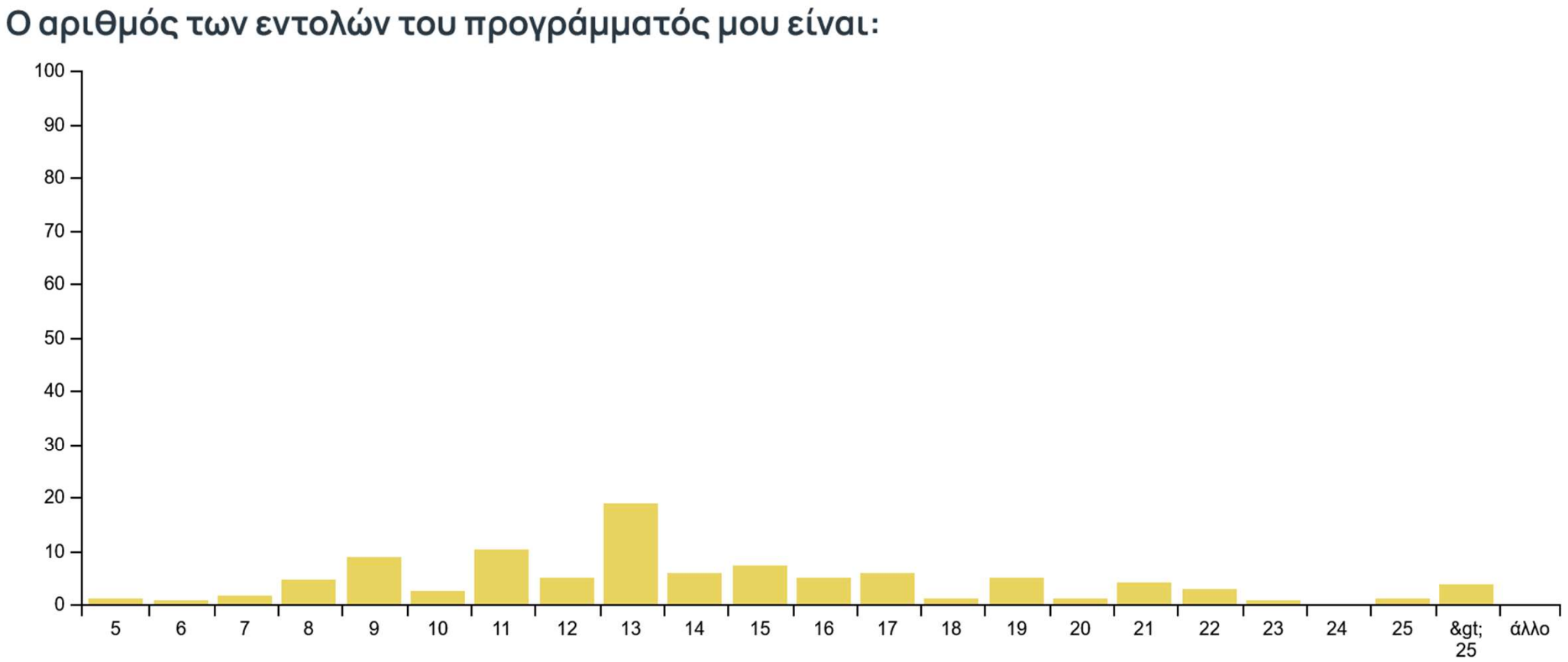
Λύση (3)

- Και μια λύση με ακόμη λιγότερες εντολές. Η συσσώρευση των αθροισμάτων μπορεί να γίνεται απευθείας στον x10 και μεταξύ των δύο βρόχων να μην μηδενίζεται κανένα γινόμενο.
- **Συνολικά μόνο 7 εντολές (υπολογίζονται και πάλι μόνο οι χρωματισμένες με βάση τις οδηγίες της εκφώνησης).**
- **Σε ποιες περιπτώσεις δεν δουλεύει σωστά το πρόγραμμα;**

Λύση (4)

```
        .text
        .globl __start
__start:
        la x3, a
        lw x6, 0(x3)
        la x3, b
        lw x7, 0(x3)
        la x3, c
        lw x8, 0(x3)
        la x3, d
        lw x9, 0(x3)
        add x10, x0, x0             # clear the product
axb:    add x10, x10, x6    # add multiplicant to product
        addi x7, x7, -1             # minus 1 the multiplier
        bgt x7, x0, axb             # continue if > zero
cxd:    add x10, x10, x8    # add multiplicant to product
        addi x9, x9, -1             # minus 1 the multiplier
        bgt x9, x0, cxd             # continue if > zero
        li  a7, 10                  # Done, terminate program
        ecall                       # au revoir...
```

Απαντήσεις (233 συμμετοχές)



Ανεβάσατε πολλές σωστές λύσεις. Συνεχίστε έτσι.

Κουίζ 7 – εκφώνηση

- (Στο κουίζ 7 το πρόβλημα είναι το ίδιο με το κουίζ 6 αλλά ο στόχος διαφορετικός.) Οι μεταβλητές a, b, c, d ενός προγράμματος που θα εκτελεστεί σε επεξεργαστή RISC-V μεταφέρονται αρχικά στους καταχωρητές x6, x7, x8, x9 (δηλαδή πρέπει να εκτελεστούν οι εντολές la και lw που φέρνουν τις τιμές τους από τη μνήμη στους καταχωρητές). Οι αριθμοί είναι απρόσημοι.
- Να γράψετε το υπόλοιπο του προγράμματος που υπολογίζει την τιμή της παράστασης a*b + c*d χωρίς να χρησιμοποιήσετε εντολές πολλαπλασιασμού ή διαίρεσης (μπορείτε να υποθέσετε ότι δεν υπάρχει ενδεχόμενο υπερχείλισης οπότε δεν χρειάζεται να κάνετε κανέναν σχετικό έλεγχο). Μπορείτε να χρησιμοποιήσετε όλες τις υπόλοιπες εντολές που μάθαμε ως τώρα και όποιους άλλους καταχωρητές θέλετε. Η τελική τιμή της παράστασης να αποθηκεύεται στον καταχωρητή x10.
- **Το πρόγραμμά σας να εκτελεί αυτή τη φορά τις λιγότερες δυνατές δυναμικές εντολές.** Στο πλήθος εντολών που εκτελούνται να υπολογίζονται όλες οι εντολές του προγράμματός σας. Για να δείτε πόσες εντολές εκτελεί το πρόγραμμά σας χρησιμοποιήστε τα εργαλεία Instruction Statistics ή Instruction Counter του RARS.
- **Ποιος είναι ο αριθμός των εντολών που εκτελεί το πρόγραμμά σας όταν οι τιμές των αριθμών είναι a=25, b=12, c=41, d=17;**

Λύση (1)

- Παρουσιάζεται ο τρόπος μέτρησης των δυναμικών εντολών (δηλαδή των εντολών που εκτελούνται) για τις δύο λύσεις του προηγούμενου Κουίζ 6.

Λύση (2) – ο πρώτος κώδικας

```
        .text
        .globl __start
__start:
        la x3, a
        lw x6, 0(x3)
        la x3, b
        lw x7, 0(x3)
        la x3, c
        lw x8, 0(x3)
        la x3, d
        lw x9, 0(x3)
        add x11, x0, x0             # clear the product
axb:    add x11, x11, x6    # add multiplicant to product
        addi x7, x7, -1             # minus 1 the multiplier
        bgt x7, x0, axb             # continue if > zero
        add x12, x0, x0             # clear the product
cxd:    add x12, x12, x8    # add multiplicant to product
        addi x9, x9, -1             # minus 1 the multiplier
        bgt x9, x0, cxd             # continue if > zero
        add x10, x11, x12
        li  a7, 10                  # Done, terminate program
        ecall                       # au revoir...
```

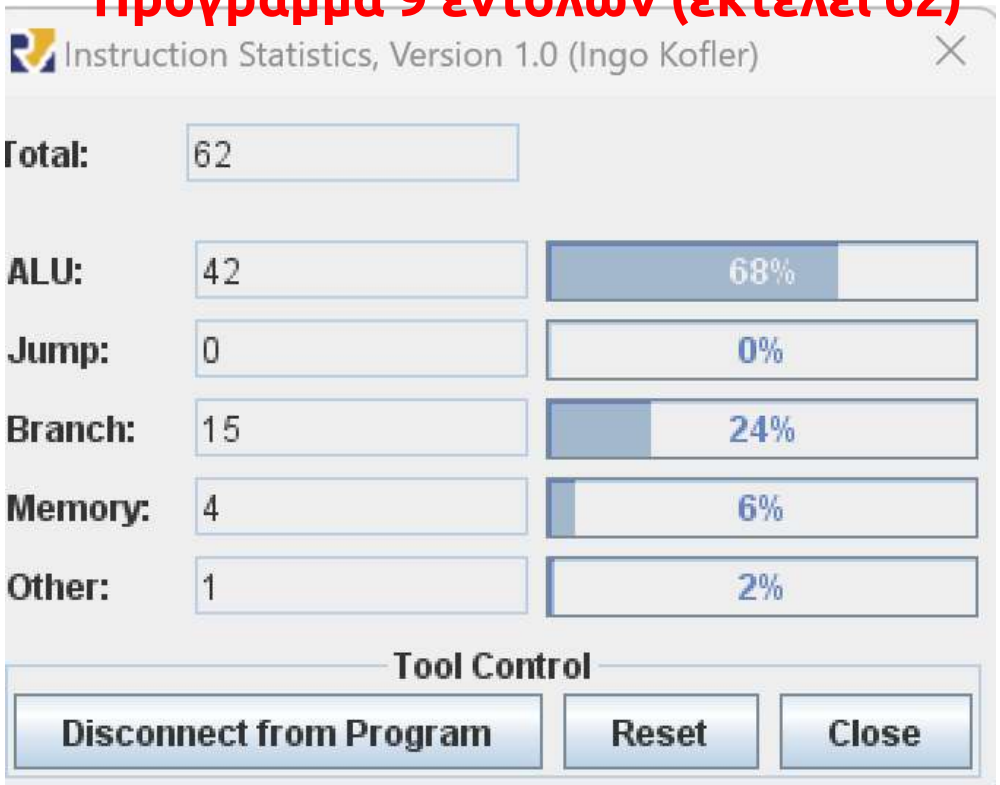
Λύση (3) – ο δεύτερος κώδικας

```
        .text
        .globl __start
__start:
        la x3, a
        lw x6, 0(x3)
        la x3, b
        lw x7, 0(x3)
        la x3, c
        lw x8, 0(x3)
        la x3, d
        lw x9, 0(x3)
        add x10, x0, x0             # clear the product
axb:    add x10, x10, x6    # add multiplicant to product
        addi x7, x7, -1             # minus 1 the multiplier
        bgt x7, x0, axb             # continue if > zero
cxd:    add x10, x10, x8    # add multiplicant to product
        addi x9, x9, -1             # minus 1 the multiplier
        bgt x9, x0, cxd             # continue if > zero
        li  a7, 10                  # Done, terminate program
        ecall                       # au revoir...
```

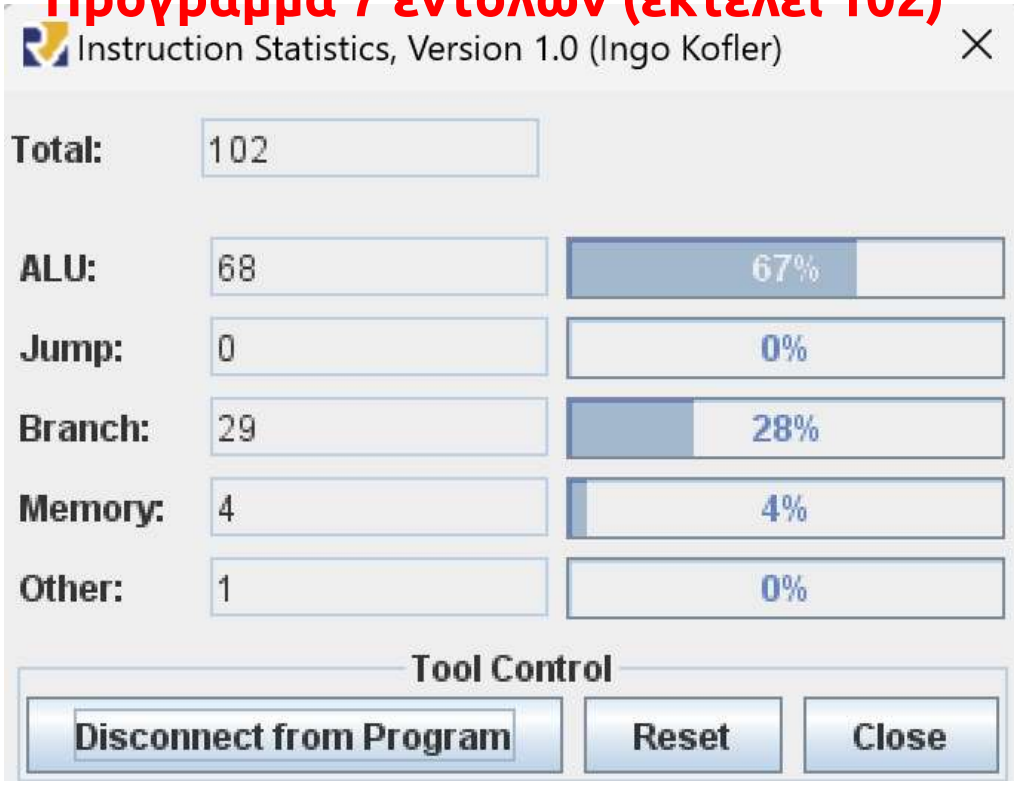
Λύση (4) – η μέτρηση εντολών

- Πριν εκτελέσουμε το πρόγραμμα μέχρι τέλους, ανοίγουμε το Tool που ονομάζεται Instruction Statistics (ή το Instruction Counter) και κάνουμε Connect to Program.
  Κατόπιν εκτελούμε το πρόγραμμα μέχρι τέλους.
  Το αποτέλεσμα για τα δύο προγράμματα είναι:

Πρόγραμμα 9 εντολών (εκτελεί 62)



Πρόγραμμα 7 εντολών (εκτελεί 102)



Απαντήσεις (191 συμμετοχές)

- Πολλές ωραίες και σωστές λύσεις

Κουίζ 8 – εκφώνηση

- Ένας μικροεπεξεργαστής RISC-V σχεδιάζεται με διαδρομή δεδομένων ενός κύκλου ρολογιού (single-cycle datapath) όπως αυτή που φαίνεται στη διαφάνεια 20 του Κεφαλαίου 4 (Εικόνα 4.11 του βιβλίου – σελίδα 317). Η εκτέλεση κάθε εντολής διαρκεί έναν κύκλο ρολογιού.
- Στον μικροεπεξεργαστή εκτελείται δύο φορές ένας βρόχος που αποτελείται από την παρακάτω ακολουθία εντολών (δεν έχει σημασία ποιοι είναι οι τελεστέοι της κάθε εντολής): `add->lw->or->sw->and->sub->beq`
- Η ALU μπορεί να εκτελέσει όλες τις απαραίτητες πράξεις (πρόσθεση, αφαίρεση, λογικό and, λογικό or, κ.λπ.)
- **Πόσες πράξεις πρόσθεσης εκτελούνται στον μικροεπεξεργαστή κατά την εκτέλεση των δύο επαναλήψεων του βρόχου;**

Λύση

- 1η προσέγγιση
  - Υποθέτουμε ότι η ALU περιέχει έναν ξεχωριστό ακέραιο αθροιστή του οποίου η έξοδος πολυπλέκεται με άλλες αριθμητικές μονάδες για να παραχθεί η έξοδος της ALU.
  - Τότε, κατά τη διάρκεια κάθε κύκλου ρολογιού στον μικροεπεξεργαστή εκτελούνται τρεις προσθέσεις. Η μία στην ALU, η δεύτερη στον αθροιστή που υπολογίζει το PC+4 και η τρίτη στον αθροιστή που υπολογίζει τη διεύθυνση της διακλάδωσης.
  - Οι δύο επαναλήψεις του βρόχου εκτελούν συνολικά 14 εντολές, **άρα θα εκτελεστούν 3x14 = 42 προσθέσεις.**
- 2η προσέγγιση
  - Υποθέτουμε ότι η ALU δεν περιέχει έναν ακέραιο αθροιστή αλλά είναι σχεδιασμένη με τέτοιο τρόπο ώστε στην έξοδο να παράγεται μόνο το επιθυμητό αποτέλεσμα.
  - Τότε κατά τη διάρκεια κάθε κύκλου ρολογιού εκτελούνται σίγουρα δύο προσθέσεις στους αθροιστές που τροφοδοτούν τον PC.
  - Στην έξοδο της ALU παράγεται το αποτέλεσμα μιας πρόσθεσης μόνο για τις εντολές add, lw, και sw.
  - **Συνολικά οι προσθέσεις είναι 2x14 + 2x3 = 34 προσθέσεις.**

Απαντήσεις (234 συμμετοχές)

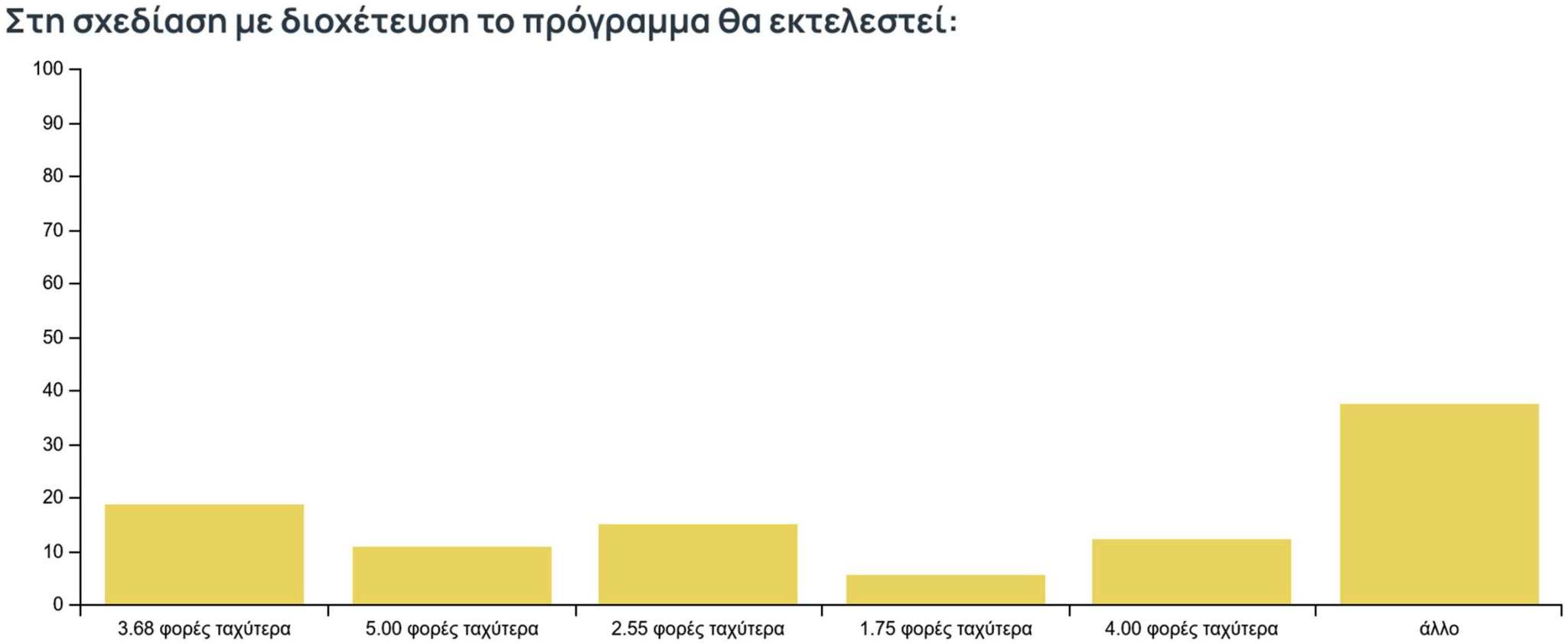
Κουίζ 9 – εκφώνηση

- Ένας μικροεπεξεργαστής RISC-V πρόκειται να σχεδιαστεί από μονάδες υλικού με τους εξής λανθάνοντες χρόνους (latencies): Καταχωρητές=90 ps, Μνήμες (εντολών και δεδομένων)=110 ps, ALU=95 ps.
- Εξετάζονται δύο εναλλακτικές σχεδιάσεις: μία με διαδρομή δεδομένων ενός κύκλου ρολογιού (single cycle datapath) και μία με ιδανική διοχέτευση (χωρίς καμία καθυστέρηση) πέντε σταδίων όπως συζητήθηκαν στο μάθημα.
- **Πόσες φορές ταχύτερα θα εκτελεστεί ένα πρόγραμμα 20 δισεκατομμυρίων δυναμικών εντολών στη σχεδίαση με διοχέτευση έναντι της σχεδίασης ενός κύκλου ρολογιού;**

Λύση

- Ο κύκλος ρολογιού στη σχεδίαση ενός κύκλου ρολογιού έχει διάρκεια T1=110 + 90 + 95 + 110 + 90 = 495 ps (ίση με το άθροισμα των χρόνων των στοιχείων του υλικού που χρησιμοποιεί η πιο αργή εντολή – μια εντολή load)
- Ο κύκλος ρολογιού στη σχεδίαση με διοχέτευση έχει διάρκεια T2=110 ps (ίση με το πιο αργό στάδιο της διοχέτευσης)
- Το πλήθος των δυναμικών εντολών που θα εκτελεστούν είναι, φυσικά, το ίδιο και στις δύο σχεδιάσεις, ενώ επίσης το CPI και των δύο σχεδιάσεων είναι 1.
- Συνεπώς η σύγκριση μπορεί να περιοριστεί στη σύγκριση της διάρκειας του κύκλου ρολογιού.
- **Η σχεδίαση με διοχέτευση είναι 495 / 110 = 4.5 φορές ταχύτερη.**

Απαντήσεις (213 συμμετοχές)



Κουίζ 10 – εκφώνηση

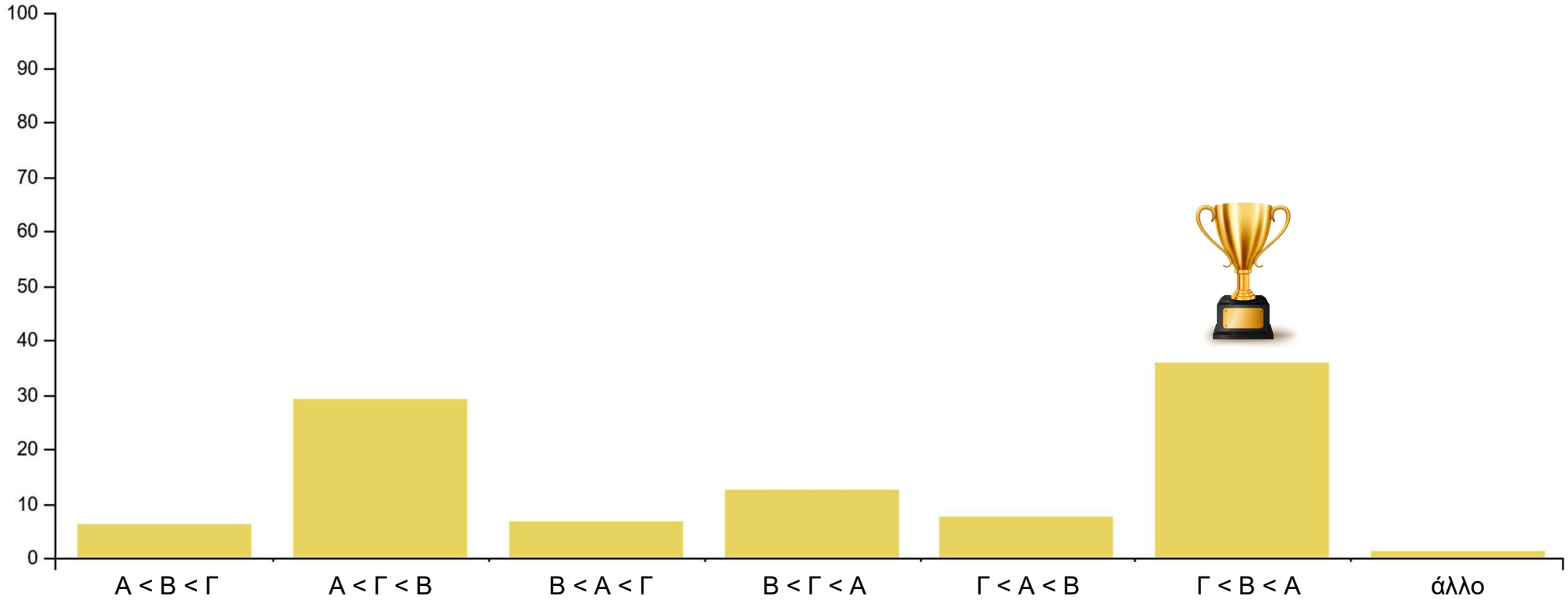
- Ένα πρόγραμμα RISC-V εκτελεί 350 τρισεκατομμύρια δυναμικές εντολές. Το πρόγραμμα αποτελείται από 15% διακλαδώσεις, 25% εντολές load που δίνουν το αποτέλεσμά τους στην αμέσως επόμενη εντολή και 30% εντολές τύπου R που δίνουν το αποτέλεσμά τους στην αμέσως επόμενη εντολή. Δεν υπάρχει καμία άλλη εξάρτηση (κίνδυνος) δομής, δεδομένων ή ελέγχου στις υπόλοιπες εντολές.
- Εκτελούμε το πρόγραμμα σε τρεις διαφορετικούς μικροεπεξεργαστές κατασκευασμένους από τα ίδια δομικά στοιχεία που έχουν τους εξής λανθάνοντες χρόνους (latencies): ALU=45 ps, Μνήμες Εντολών και Δεδομένων=60 ps, και Καταχωρητές=50 ps. Αγνοείστε τους λανθάνοντες χρόνους οποιουδήποτε άλλου στοιχείου.
- Ο μικροεπεξεργαστής Α έχει διαδρομή δεδομένων ενός κύκλου ρολογιού (single cycle datapath) και δαπανά ηλεκτρική ισχύ 1 watt.
- Ο μικροεπεξεργαστής Β έχει διαδρομή δεδομένων με διοχέτευση (pipeline). Το υλικό του ανιχνεύει τους κινδύνους δεδομένων και ελέγχου και προσθέτει φυσαλίδες (bubbles) ή καθυστερήσεις (stalls) και η επίλυση των διακλαδώσεων (σύγκριση καταχωρητών) γίνεται στο στάδιο EX (στην ALU). Δαπανά ηλεκτρική ισχύ 1.2 watt.
- Ο μικροεπεξεργαστής Γ έχει διαδρομή δεδομένων με διοχέτευση, πλήρη προώθηση (forwarding), επίλυση διακλαδώσεων επίσης στο στάδιο EX, και διαθέτει πρόβλεψη διακλάδωσης (branch prediction) που προβλέπει σωστά το 90% των διακλαδώσεων. Δαπανά ηλεκτρική ισχύ 1.3 watt.
- **Κατατάξτε τους τρεις μικροεπεξεργαστές σε αύξουσα σειρά κατανάλωσης ηλεκτρικής ενέργειας για την εκτέλεση του προγράμματος.**

Λύση

- CPI(A) = 1.00 (διαδρομή δεδομένων ενός κύκλου)
- CPI(B) = 1 + 0.25*2 + 0.30*2 + 0.15*2 = 2.40
- (2 κύκλοι καθυστέρησης μεταξύ εξαρτώμενων εντολών και των τριών τύπων)
- CPI(Γ) = 1 + 0.25*1 + 0.30*0 + 0.15*0.10*2 = 1.28
- (1 μόνο κύκλος καθυστέρησης για load-use κινδύνους και 0 κύκλοι μεταξύ τύπου R και επόμενης, η ποινή των 2 κύκλων καθυστέρησης για τις διακλαδώσεις δαπανάται μόνο για το 10% των διακλαδώσεων που δεν προβλέπονται σωστά)
- CLK(A) = 60 + 50 + 45 + 60 + 50 ps = 265 ps
- CLK(B) = MAX{60, 50, 45, 60, 50} = 60 ps
- CLK(Γ) = MAX{60, 50, 45, 60, 50} = 60 ps
- $T(A) = I * CPI(A) * CLK(A) = 350*10^{12} * 1 * 265*10^{-12}$ s = 92 750 s
- $T(B) = I * CPI(B) * CLK(B) = 350*10^{12} * 2.40 * 60*10^{-12}$ s = 50 400 s
- $T(Γ) = I * CPI(Γ) * CLK(Γ) = 350*10^{12} * 1.28 * 60*10^{-12}$ s = 26 880 s
- Energy(A) = T(A) * Power(A) = 92 750 s * 1 watt = 92 750 joules
- Energy(B) = T(B) * Power(B) = 50 400 s * 1.2 watt = 60 480 joules
- Energy(Γ) = T(Γ) * Power(Γ) = 26 880 s * 1.3 watt = 34 944 joules
- **Άρα Energy(Γ) < Energy(B) < Energy(A)**

Απαντήσεις (223 συμμετοχές)

Η αύξουσα σειρά κατανάλωσης ενέργειας των επεξεργαστών είναι:



Κουίζ 11 – εκφώνηση

- Ένα πρόγραμμα Π θα εκτελεστεί σε πέντε διαφορετικούς μικροεπεξεργαστές RISC-V με ονόματα Υ1, Υ2, Υ3, Υ4, Υ5. Κάθε μικροεπεξεργαστής διαθέτει τον δικό του μεταγλωττιστή C1, C2, C3, C4, C5, αντίστοιχα που παράγει εκτελέσιμο κώδικα για το Π που εκτελεί 1.00 δις, 1.10 δις, 1.05 δις, 1.25 δις, 1.17 δις δυναμικές εντολές, αντίστοιχα.
- Οι βασικές μονάδες του υλικού έχουν λανθάνοντες χρόνους (για όλους του επεξεργαστές): Μνήμες=75 ps, ALU=65 ps, Καταχωρητές=55 ps.
- Ο Υ1 διαθέτει απλή σχεδίαση ενός κύκλου ρολογιού.
- Ο Υ2 διαθέτει σχεδίαση πολλών κύκλων ρολογιού με CPI=3 για τις διακλαδώσεις (που αποτελούν το 10% των εντολών), CPI=4 για τις αποθηκεύσεις και τις εντολές τύπου-R (που αποτελούν μαζί το 75% των εντολών) και CPI=5 για τις εντολές φόρτωσης (που είναι οι υπόλοιπες).
- Ο Υ3 διαθέτει διοχέτευση πέντε σταδίων χωρίς προώθηση και μόνο ανιχνεύει τους κινδύνους προσθέτοντας καθυστερήσεις (stalls). Το 45% των εντολών επιβαρύνονται με έναν επιπλέον κύκλο καθυστέρησης λόγω κινδύνων δεδομένων και το 13% με δύο κύκλους καθυστέρησης λόγω κινδύνων ελέγχου.
- Ο Υ4 διαθέτει διοχέτευση πέντε σταδίων με πλήρη προώθηση και γι' αυτό μόνο το 20% των εντολών επιβαρύνονται με έναν επιπλέον κύκλο καθυστέρησης λόγω κινδύνων δεδομένων. Επειδή διαθέτει και πολύ καλή μονάδα πρόβλεψης διακλάδωσης μόνο το 3% των εντολών επιβαρύνονται με δύο κύκλους καθυστέρησης λόγω κινδύνων ελέγχου.
- Ο Υ5 είναι μια «μαγική» σχεδίαση που εξαφανίζει όλους τους κινδύνους της διοχέτευσης η οποία λειτουργεί ιδανικά.
- Ο Υ1 κοστίζει 10 ευρώ, ο Υ2 κοστίζει 12 ευρώ, ο Υ3 κοστίζει 18 ευρώ, ο Υ4 κοστίζει 25 ευρώ, και ο Υ5 κοστίζει 40 ευρώ.
- **Ποιος από τους πέντε μικροεπεξεργαστές δίνει τον καλύτερο λόγο απόδοσης προς κόστος (performance per cost ratio) για το πρόγραμμα Π;**

Λύση (1)

- Για κάθε μικροεπεξεργαστή τα πλήθη των δυναμικών εντολών δίνονται από την εκφώνηση.
- Το CPI του καθενός υπολογίζεται ως εξής:
  - CPI-Y1 = **1.00** (πρόκειται για CPU ενός κύκλου ρολογιού)
  - CPI-Y2 = 0.10*3+0.75*4+0.15*5 = **4.05** (πρόκειται για CPU πολλών κύκλων ρολογιού στην οποία κάθε κατηγορία εντολών διαρκεί 3, 4 ή 5 κύκλους)
  - CPU-Y3 = 1+0.45*1+0.13*2 =**1.71** (πρόκειται για CPU με διοχέτευση στην οποία επιβαρύνεται το ιδανικό CPI=1 με επιπλέον κύκλους λόγω κινδύνων δεδομένων και ελέγχου)
  - CPI-Y4 = 1+0.2*1+0.03*2 = **1.26** (ομοίως αλλά με λιγότερες καθυστερήσεις αφού είναι πιο προηγμένη σχεδίαση με διοχέτευση)
  - CPI-Y5 = **1.00** (καμία επιβάρυνση λόγω κινδύνων, η τέλεια διοχέτευση)

Λύση (2)

- Η διάρκεια του κύκλου ρολογιού για κάθε σχεδίαση υπολογίζεται ως εξής:
  - Clock-cycle-Y1 = (75+55+65+75+55)*10^{-12} s (πρόκειται για CPU ενός κύκλου ρολογιού και το ρολόι έχει διάρκεια όσο η πιο αργή εντολή, load, δηλαδή το άθροισμα των χρόνων των πέντε τμημάτων του υλικού)
  - Clock-cycle-Y2 = max{75, 55, 65, 75, 55} * 10^{-12} s =75*10^{-12} s (στη CPU πολλών κύκλων ρολογιού η διάρκεια του ρολογιού είναι ίση με το πιο αργό από τα πέντε στάδια του υλικού)
  - Clock-cycle-Y3 = max{75, 55, 65, 75, 55} * 10^{-12} s = 75*10^{-12} s (στη CPU με διοχέτευση η διάρκεια του ρολογιού είναι ίση με το πιο αργό από τα πέντε στάδια του υλικού)
  - Clock-cycle-Y4 = 75*10^{-12} s (ομοίως)
  - Clock-cycle-Y5 = 75*10^{-12} s (ομοίως)
- Με χρήση της βασικής εξίσωσης της απόδοσης
  Time = Instructions * CPI * Clock-cycle
- **Υπολογίζουμε τους χρόνους εκτέλεσης του Π στους πέντε μικροεπεξεργαστές**
- **Time-Y1 = 0.3250 s, Time-Y2 = 0.3341 s, Time-Y3=0.1347 s, Time-Y4 = 0.1181 s, Time-Y5 = 0.0878 s**

Λύση (3)

- Ο ζητούμενος λόγος είναι:
- Λ = Performance/Cost ή Λ = 1 / (Time*Cost)
- **Συνεπώς**
- **Λ-Υ1 = 0.31**
- **Λ-Υ2 = 0.25**
- **Λ-Υ3 = 0.41**
- **Λ-Υ4 = 0.34**
- **Λ-Υ5 = 0.28**

- **Νικητής είναι ο μικροεπεξεργαστής Υ3 που παρέχει τον υψηλότερο λόγο Λ.**

Απαντήσεις (232 συμμετοχές)