ΕΘΝΙΚΟ & ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
ΜΑΘΗΜΑ: **ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΥΠΟΛΟΓΙΣΤΩΝ ΙΙ**

Εργασία 2 (υποχρεωτική) – Κρυφές Μνήμες

ΑΚΑΔΗΜΑΪΚΟ ΕΤΟΣ 2021 – 2022

(ΕΚΦΩΝΗΣΗ) ΔΕΥΤΕΡΑ 20 ΔΕΚΕΜΒΡΙΟΥ 2021

(ΠΑΡΑΔΟΣΗ ΣΤΟ ECLASS ΜΕΧΡΙ) **ΠΑΡΑΣΚΕΥΗ 21 ΙΑΝΟΥΑΡΙΟΥ 2022**

Επώνυμο	Όνομα	Αριθμός Μητρώου	Email
Γιαννοπούλου	Κατερίνα	1115201600034	sdi1600034@di.uoa.gr
Φώτης	Γιάννης	1115201700182	sdi1700182@di.uoa.gr

Πληροφορίες για τις Υποχρεωτικές Εργασίες του μαθήματος

- Οι υποχρεωτικές εργασίες του μαθήματος είναι **δύο**. Σκοπός τους είναι η κατανόηση των εννοιών του μαθήματος με χρήση αρχιτεκτονικών προσομοιωτών. Η πρώτη υποχρεωτική εργασία αφορά τη διοχέτευση (pipelining) και η δεύτερη (αυτή) αφορά τις κρυφές μνήμες (cache memories).
- Οι δύο εργασίες είναι υποχρεωτικές και η βαθμολογία του μαθήματος θα προκύπτει από το γραπτό (60%), την εργασία της διοχέτευσης (20%), και την εργασία των κρυφών μνημών (20%).
- Κάθε ομάδα μπορεί να αποτελείται **από 1 έως και 3 φοιτητές** (η υποβολή να γίνει μόνο από έναν φοιτητή εκ μέρους όλους της ομάδας). Συμπληρώστε τα στοιχεία όλων των μελών της ομάδας στον παραπάνω πίνακα. Όλα τα μέλη της ομάδας πρέπει να έχουν ισότιμη συμμετοχή και να γνωρίζουν τις λεπτομέρειες της υλοποίησης της ομάδας.
- Για την εξεταστική Σεπτεμβρίου δε θα δοθούν άλλες εργασίες. Το Σεπτέμβριο εξετάζεται μόνο το γραπτό.
- Σε περίπτωση αντιγραφής θα μηδενίζονται όλες οι ομάδες που μετέχουν σε αυτή.
- Η παράδοση της **Εργασίας Κρυφών Μνημών** πρέπει να γίνει **ηλεκτρονικά μέχρι τα μεσάνυχτα της προθεσμίας και μόνο στο eclass** (να ανεβάσετε ένα μόνο αρχείο zip ή rar με την τεκμηρίωσή σας σε PDF και τους κώδικές σας). **Μην περιμένετε μέχρι την τελευταία στιγμή – κάθε εργασία απαιτεί τον χρόνο της**.

Ζητούμενο

Το ζητούμενο της εργασίας είναι η **σχεδίαση μιας «οικογένειας» τριών μικροεπεξεργαστών** που να διαφέρουν στην απόδοση και το κόστος για την ίδια υπολογιστική εργασία. Η σχεδίαση και η αξιολόγηση των επεξεργαστών θα γίνει στον προσομοιωτή QtMips.

Η υπολογιστική εργασία αποτελείται από δύο διακριτά βήματα (sort and search):

(α) Ένας αρχικά μη ταξινομημένος πίνακας 40 000 θετικών ακεραίων αριθμών (n[1], … , n[40 000]) μεγέθους 1 byte ο καθένας πρέπει να ταξινομηθεί με αύξουσα σειρά. Πρέπει να γίνεται έλεγχος ότι οι αριθμοί είναι πράγματι θετικοί και το πρόγραμμα να σταματάει εάν βρεθεί αρνητικός αριθμός ή το μηδέν. Ο πίνακας βρίσκεται στο τμήμα δεδομένων του προγράμματός σας και δεν απαιτείται είσοδος/έξοδος. Η ταξινόμηση να χρησιμοποιεί τον αλγόριθμο quicksort.

(β) Στον ταξινομημένο πίνακα των ακεραίων του 1 byte γίνεται αναζήτηση ενός συγκεκριμένου αριθμού X (που επίσης βρίσκεται στο τμήμα δεδομένων του προγράμματός σας). Η αναζήτηση να χρησιμοποιεί τον αλγόριθμο δυαδικής αναζήτησης (binary search). Αν ο αριθμός βρεθεί τότε μια μεταβλητή POS (και πάλι στο τμήμα δεδομένων του προγράμματος) πρέπει να περιέχει την πρώτη θέση στην οποία βρέθηκε ο αριθμός (1 μέχρι 40 000). Αν ο αριθμός δεν βρεθεί τότε η POS πρέπει να έχει την τιμή 0.

Οι τρεις επεξεργαστές έχουν κοινή σχεδίαση στον πυρήνα του pipeline: διαθέτουν branch predictor των 2-bit με BHT που προσπελαύνεται με 5 bit και επίλυση των διακλαδώσεων στο στάδιο EX και πλήρη μονάδα κινδύνων με προώθηση. Διαφέρουν στο σύστημα της μνήμης.

Ο πρώτος επεξεργαστής (codename: *turtle*) είναι μια φθηνή έκδοση της οικογένειας με το ελάχιστο δυνατό κόστος που σημαίνει ότι δεν χρησιμοποιεί κρυφή μνήμη (cache). Η προσπέλαση της κύριας μνήμης διαρκεί 40 κύκλους ρολογιού. Το κόστος του επεξεργαστή είναι 20 ευρώ και ο ρυθμός του ρολογιού του είναι 500 MHz.

Ο δεύτερος επεξεργαστής (codename: *rabbit*) είναι μια μέση έκδοση της οικογένειας που χρησιμοποιεί μόνο ένα επίπεδο κρυφής μνήμης (L1) για εντολές και δεδομένα. Η προσπέλαση της κύριας μνήμης και σε αυτόν τον επεξεργαστή διαρκεί 40 κύκλους. Η κρυφή μνήμη εντολών/προγράμματος (L1 program cache στην ορολογία του QtMips) έχει μέγεθος 8KB και μπορεί να περιέχει 4, 8, 16, ή 32 λέξεις ανά μπλοκ. Μπορείτε να επιλέξετε όποια συσχετιστικότητα (associativity) και όποια πολιτική αντικατάστασης επιθυμείτε. Η κρυφή μνήμη δεδομένων (L1 data cache) μπορεί να έχει μέγεθος μεταξύ 4, 8, ή 16KB και μπορεί να περιέχει 4, 8, 16, ή 32 λέξεις ανά μπλοκ. Μπορείτε και πάλι να επιλέξετε όποια συσχετιστικότητα και όποια πολιτική αντικατάστασης επιθυμείτε. Η πολιτική εγγραφής και κατανομής πρέπει να είναι ετερόχρονη (write back) και με κατανομή σε εγγραφή (write allocate). Το κόστος αυτού του επεξεργαστή είναι 20, 25, ή 30 ευρώ υψηλότερο από τον προηγούμενο αν η κρυφή μνήμη δεδομένων έχει μέγεθος 4, 8, ή 16KB, αντίστοιχα. Ο ρυθμός ρολογιού είναι 500 MHz αν η συσχετιστικότητα είναι 1 (direct mapped) αλλά μειώνεται κατά 10 MHz για κάθε διπλασιασμό της συσχετιστικότητας.

Ο τρίτος επεξεργαστής (codename: *puma*) είναι μια προηγμένη έκδοση της οικογένειας που χρησιμοποιεί δύο επίπεδα κρυφής μνήμης (ξεχωριστή L1 για εντολές και δεδομένα και ενιαία L2). Η προσπέλαση της κύριας μνήμης και σε αυτόν τον επεξεργαστή διαρκεί 40 κύκλους και η προσπέλαση της L2 διαρκεί 5 κύκλους. Οι κρυφές μνήμες L1 είναι ίδιες με του προηγούμενου επεξεργαστή (θα κρατήσετε τη σχεδίαση στην οποία καταλήξατε – συνεπώς και το κόστος και τον ρυθμό ρολογιού). Η κρυφή μνήμη L2 έχει μέγεθος 16, 32, ή 64 KB και έχει ίδιο μέγεθος μπλοκ με την L1 data cache του προηγούμενου επεξεργαστή. Μπορείτε να επιλέξετε όποια συσχετιστικότητα και όποια πολιτική αντικατάστασης επιθυμείτε για την L2. Η πολιτική εγγραφής και κατανομής πρέπει να είναι ετερόχρονη (write back) και με κατανομή σε εγγραφή (write allocate). Το κόστος αυτού του επεξεργαστή είναι 50, 75 ή 100 ευρώ υψηλότερο από τον προηγούμενο αν η κρυφή μνήμη L2 έχει μέγεθος 16, 32, ή 64 KB, αντίστοιχα. Ο ρυθμός ρολογιού είναι ίσος με αυτόν του προηγούμενου επεξεργαστή.

- Να συμπληρώσετε τον παρακάτω πίνακα με **τα χαρακτηριστικά της σχεδίασής σας** για τον καθένα από τους τρεις επεξεργαστές.
- Να μετρήσετε **τον χρόνο εκτέλεσης** του προγράμματός σας στις τρεις σχεδιάσεις σας χρησιμοποιώντας τον αριθμό των κύκλων ρολογιού από τον QtMips και τον ρυθμό ρολογιού της σχεδίασής σας. Εάν χρησιμοποιήσετε διάφορα σύνολα δεδομένων να περιγράψετε τον τρόπο με τον οποίο παίρνετε τις μετρήσεις σας (μπορείτε να χρησιμοποιήσετε μέσες τιμές για παράδειγμα).
- Να υπολογίσετε **τον λόγο απόδοσης προς κόστος** του προγράμματός σας και στις τρεις σχεδιάσεις σας (όπως κάνατε τον υπολογισμό και στην Εργασία 1).
- Να περιγράψετε **τρία διαφορετικά υπολογιστικά συστήματα** στα οποία θα χρησιμοποιούσατε καθέναν από τους τρεις επεξεργαστές. Ποιοι είναι οι λόγοι για τους οποίους θεωρείτε κατάλληλο τον καθένα για το συγκεκριμένο σύστημα.

Χαρακτηριστικό	**Πρώτος επεξεργαστής (*turtle*)**	**Δεύτερος επεξεργαστής (*rabbit*)**	**Τρίτος επεξεργαστής (*puma*)**
L1 program cache (μέγεθος, μέγεθος μπλοκ, συσχετιστικότητα, πολιτική αντικατάστασης)			
L1 data cache (μέγεθος, μέγεθος μπλοκ, συσχετιστικότητα, πολιτική αντικατάστασης)			
L2 cache (μέγεθος, μέγεθος μπλοκ, συσχετιστικότητα, πολιτική αντικατάστασης)			
Κόστος			
Ρυθμός ρολογιού			
Κύκλοι εκτέλεσης			
Χρόνος εκτέλεσης			
Λόγος απόδοσης/κόστους			
Σύστημα για το οποίο είναι κατάλληλος (~150 λέξεις)			

Τεκμηρίωση

[Σύντομη τεκμηρίωση της λύσης σας μέχρι **10 σελίδες ξεκινώντας από την επόμενη σελίδα** – μην αλλάζετε τη μορφοποίηση του κειμένου (**και παραδώστε την τεκμηρίωση σε αρχείο PDF**). Η τεκμηρίωσή σας πρέπει να περιλαμβάνει παραδείγματα ορθής εκτέλεσης και σχολιασμό για την επίλυση του προβλήματος και την επίτευξη του ζητούμενου. Μπορείτε να χρησιμοποιήσετε εικόνες, διαγράμματα και ό,τι άλλο μπορεί να βοηθήσει στην εξήγηση της δουλειάς σας.]

Εισαγωγή:

Ο κώδικάς μας είναι στο αρχείο ca-II-handout-2.s. Έχουμε συμπεριλάβει επιπλέον και τις ενδεικτικές τιμές του πίνακα των 40.000 ακεραίων που χρησιμοποιήσαμε στο αρχείο Dataset.txt. Το dataset αυτό περιέχει μόνο αριθμούς στο σύνολο [1,127], ωστόσο έχουμε ελέγξει ότι το πρόγραμμά μας σταματά αν εντοπίσει μη έγκυρο αριθμό σε οποιαδήποτε θέση. Η εργασία μας αναπτύχθηκε και δοκιμάστηκε στην έκδοση qt 5.12.9 v0.94 του QtMips.

Έλεγχος input:

Αποφασίσαμε να ελέγχουμε την εγκυρότητα των δεδομένων με ένα γραμμικό scan όλου του πίνακα στην αρχή του προγράμματος. Η λύση αυτή προτιμήθηκε έναντι του ελέγχου μέσα στην ίδια την quicksort λόγω του μεγέθους του πίνακα. Συγκεκριμένα, παρόλο που με μία μόνο εντολή μπορούμε μέσα στην quicksort να ελέγξουμε έναν αριθμό, λόγω της πολυπλοκότητας O(nlogn), πολλοί αριθμοί ελέγχονται 10 φορές κατά μέσο όρο, ενώ για την περίπτωσή μας αρκούν 5 εντολές για κάθε έναν. Αν το πρόβλημα ήταν μικρότερο, θα επιλέγαμε τη δεύτερη μέθοδο.

Σχεδιαστικές επιλογές – ανάπτυξη κώδικα:

Ο αλγόριθμος που υλοποιήσαμε για την ταξινόμηση του πίνακα είναι μια παραλλαγή της quicksort, και συγκεκριμένα διαφέρει από τον κλασσικό αλγόριθμο μόνο στη διαδικασία του partitioning. Ο λόγος που επιλέξαμε αυτή την εκδοχή της quicksort, είναι γιατί αντιμετωπίζει πολύ αποτελεσματικά την ύπαρξη διπλότυπων αριθμών, κάτι το οποίο συμβαίνει σε μεγάλο βαθμό, αφού το εύρος τιμών που έχουμε είναι πολύ μικρό σε σχέση με το μέγεθος του πίνακα. Το συγκεκριμένο φαινόμενο ονομάζεται Dutch National Flag Problem και έχει αναλυθεί από τον Edsger W. Dijkstra.

Παρ' όλα αυτά, αρχικά κάναμε μετρήσεις και με την κλασσική quicksort, η οποία χρειαζόταν περίπου 49 εκατομμύρια εντολές. Η έκδοση που σας παραδίδουμε λύνει το ίδιο πρόβλημα με μόλις 3,4 εκατομμύρια εντολές, επομένως βλέπουμε μία βελτίωση κατά 14,4 φορές μόνο από την τροποποίηση του κώδικα.

Τέλος θα θέλαμε να επισημάνουμε ότι έχουμε κάνει κάποιες επιλογές στον κώδικα, οι οποίες επιβαρύνουν λίγο το συνολικό αριθμό εντολών, αλλά μειώνουν δραστικά τα data hazards. Για παράδειγμα, στη μέθοδο partition της quicksort και συγκεκριμένα στο βρόγχο while, κάνουμε lb στον καταχωρητή, τον αριθμό που θέλουμε να συγκρίνουμε με το pivot πριν να ελεγχθεί η κεντρική συνθήκη που τερματίζει την partition. Αυτό έχει ως αποτέλεσμα να κάνουμε μία περιττή εντολή load τη στιγμή που πρέπει να τερματίσει η partition. Γνωρίζουμε ότι αυτό συμβαίνει μόνο μία φορά κατά την κλήση της quicksort. Στην περίπτωση του επεξεργαστή turtle, ειδικά αν ο αριθμός αυτός δε βρίσκεται στον επεξεργαστή, οι συνολικοί κύκλοι επιβαρύνονται. Ωστόσο, στις άλλες δύο περιπτώσεις, επειδή η ποινή του data stall διαρκεί πολύ λιγότερους κύκλους τις περισσότερες φορές, η επιβάρυνση αυτή είναι πολύ μικρότερη. Το κέρδος που έχουμε από αυτή την επιλογή, είναι ότι φορτώνουμε τον αριθμό αυτό πολύ νωρίτερα από ότι θα χρειαστεί και έτσι αποφεύγονται τα data stalls που συμβαίνουν όσο τρέχει η partition. Αυτό συμβαίνει πολύ περισσότερες φορές από τη μία που θα εκτελέσουμε την load χωρίς να χρειάζεται και επομένως οι συνολικά κερδισμένοι κύκλοι στους επεξεργαστές rabbit και puma είναι αρκετά περισσότεροι από αυτούς που χάνονται. Ενδεικτικά σε δοκιμές που κάναμε στον επεξεργαστή rabbit, το CPI με τη load μετά τη συνθήκη branch ήταν 1,083 έναντι του 1,089 στην τελική μας υλοποίηση. Ωστόσο οι συνολικοί κύκλοι που εκτελέσθηκαν στην τελευταία ήταν λιγότεροι, με αποτέλεσμα ο λόγος απόδοσης προς κόστος να είναι καλύτερο κατά 2%. Δεδομένου ότι το κύριο μέτρο σύγκρισης στην εργασία μας είναι ο λόγος αυτός, προχωρήσαμε με την παραπάνω λύση.

Επιλογή μεταξύ των διαφορετικών configuration για κάθε επεξεργαστή:

1) **Επεξεργαστής Turtle:**

Για το συγκεκριμένο επεξεργαστή υπάρχει μόνο ένα configuration. Βάζοντας τις ρυθμίσεις που ζητούνται, τα αποτελέσματα είναι:

Cost	Clock	Instructions	Total Cycles	CPI	Time	Value
20€	500MHz	3.310.150	179.333.872	54,177	0,35867 s	0,139404785728 * 1/(€*s)

2) **Επεξεργαστής Rabbit:**

Επειδή οι επιλογές για τη διαμόρφωση της cache είναι πολλές, δεν είναι δυνατόν να ελέγξουμε όλους τους συνδυασμούς των configurations. Επομένως προβήκαμε σε μία σειρά από αποφάσεις με βάση όσα γνωρίζουμε από τη θεωρία για να μειώσουμε σημαντικά το πλήθος των περιπτώσεων που πρέπει να εξεταστούν.

Όσο αφορά την level 1 program cache, δεν έχει νόημα να εξετάσουμε πολλές περιπτώσεις, μιας και οι εντολές του προγράμματός μας είναι λιγότερες από 100, που σημαίνει ότι αρκούν λιγότερα από 400 Bytes cache ώστε να μην συμβεί κανένα cache miss πέρα από τα αρχικά στην εκκίνηση του προγράμματος. Με βάση αυτά τα μεγέθη, συμπεραίνουμε ότι ακόμη και 0,5kB αρκούν για την program cache χωρίς να μειώνεται το όφελος που έχουμε από

αυτή. Επιπλέον, θεωρούμε ότι η χρήση συσχετιστικότητας για την program cache αποτελεί σπατάλη, δεδομένου ότι δε θα γεμίσει ποτέ πλήρως με εντολές και δε θα γίνει καμία αντικατάσταση. Τέλος, το μέγεθος του block, επηρεάζει τα αρχικά misses, αφού με μέγεθος 32, μετά από μόλις 3 misses θα έρθουν όλες οι εντολές (δηλαδή 120 κύκλοι), ενώ αν επιλέξουμε μικρότερο μέγεθος θα συμβούν έως και 8 φορές περισσότερα. Ακόμη και έτσι, το ποσοστό των misses της program cache έχουν μηδαμινό αντίκτυπο στους συνολικούς κύκλους του προγράμματος, επομένως δεν έχει ουσιαστική διαφορά οποιοδήποτε μέγεθος block και αν επιλέξουμε. Επειδή όμως δεν υπάρχει κάποια επιβάρυνση, επιλέξαμε η τελική διαμόρφωση της level 1 cache να έχει 64 sets, μέγεθος block 32 words και είναι direct-mapped.

Για την level 1 data cache πρέπει να λάβουμε υπόψιν περισσότερες παραμέτρους. Αρχικά, οι διαφορές μεταξύ 4, 8 και 16 kB περιμένουμε να επηρεάσουν πολύ το χρόνο εκτέλεσης, αλλά λόγω των διαφορετικών τιμών τους δεν είναι εύκολο να υποθέσουμε ποια από τις τρεις συμφέρει περισσότερο. Γι' αυτό θα δοκιμάσουμε μία υλοποίηση από κάθε μέγεθος για να αποφασίσουμε. Ακόμη και έτσι όμως πρέπει να περιορίσουμε τις επιλογές μας και να εξετάσουμε μόνο τις καλύτερες υλοποιήσεις από κάθε μέγεθος data cache.

Για την επιλογή του Block size σκεφτήκαμε ότι αφενός το γραμμικό πέρασμα του πίνακα στην αρχή του προγράμματος και αφετέρου οι πολλαπλές συγκρίσεις και αντικαταστάσεις γειτονικών στοιχείων του πίνακα που κάνει η quicksort επωφελούνται από το μεγάλο μέγεθος block. Επομένως, επιλέξαμε τις 32 λέξεις επειδή δεν υπάρχει κάποια επιβάρυνση και θεωρούμε ότι το όφελος που έχουμε από την αξιοποίηση του spacial-locality είναι πολύ μεγαλύτερο από το ενδεχόμενο cache-pollution.

Γνωρίζουμε ότι το set-associativity βοηθά περισσότερο τις μικρές cache, μιας και εκεί ο ανταγωνισμός των block της μνήμης είναι μεγαλύτερος ανά set και πολλές φορές αν έχουμε direct-mapped cache μπορεί να αντικατασταθούν δεδομένα που χρειάζονται στο άμεσο μέλλον. Επομένως περιμένουμε το set-associativity να προσφέρει μεγαλύτερο όφελος στην cache των 4kB, λιγότερο στα 8kB και ακόμη λιγότερο στα 16kB. Επειδή ο ρυθμός ρολογιού επηρεάζεται αρνητικά, προφανώς δεν έχει νόημα να αυξάνουμε τη συσχετιστικότητα επ' αόριστον.

Ξεκινήσαμε κάνοντας κάποιες δοκιμές για data-cache 4kB:

- Number of sets = 32, Block size: = 32, Set associativity = 1

Cost	Clock	Instructions	Total Cycles	CPI	Time	Value
40€	500MHz	3.424.091	3.908.412	1,141	0,00782 s	3,19822986931 * 1/(€*s)

- Number of sets = 16, Block size: = 32, Set associativity = 2

Cost	Clock	Instructions	Total Cycles	CPI	Time	Value
40€	490MHz	3.424.091	3.728.972	1,089	0,00761 s	3,28508768636 * 1/(€*s)

- Number of sets = 8, Block size: = 32, Set associativity = 4

Cost	Clock	Instructions	Total Cycles	CPI	Time	Value
40€	480MHz	3.424.091	3.729.292	1,089	0,00777 s	3,21776894917 * 1/(€*s)

Παρατηρούμε ότι συνολικά η δεύτερη περίπτωση έχει την καλύτερη απόδοση, τόσο σε χρόνο όσο και σε λόγο απόδοσης προς χρόνο, επομένως για τα 4kB συμφέρει να έχουμε 2-way cache association. Έπειτα ελέγχουμε αν το μέγεθος block που διαλέξαμε είναι το βέλτιστο, μόνο για την καλύτερη απ' τις παραπάνω υλοποιήσεις:

- Number of sets = 32, Block size: = 16, Set associativity = 2

Cost	Clock	Instructions	Total Cycles	CPI	Time	Value
45€	490MHz	3.424.091	3.801.499	1,110	0,00776 s	3,222413053377 * 1/(€*s)

Όπως φαίνεται, το κέρδος από την αξιοποίηση του spacial locality της cache, είναι μεγαλύτερο από το cache pollution και έτσι η επιλογή μεγέθους block 32 words είναι η καλύτερη.

Συνεχίζοντας τη μελέτη για τη μεγαλύτερη cache των 8kB, περιμένουμε το πιο συμφέρον set associativity να είναι το πολύ 2, αφού επωφελείται λιγότερο από τη συσχετιστικότητα. Με βάση τον παραπάνω συλλογισμό, εξετάσαμε τις παρακάτω περιπτώσεις:

- Number of sets = 64, Block size: = 32, Set associativity = 1

Cost	Clock	Instructions	Total Cycles	CPI	Time	Value
45€	500MHz	3.424.091	3.773.292	1,102	0,00755 s	2,94467300996 * 1/(€*s)

- Number of sets = 32, Block size: = 32, Set associativity = 2

Cost	Clock	Instructions	Total Cycles	CPI	Time	Value
45€	490MHz	3.424.091	3.707.892	1,083	0,00757 s	2,93667908582 * 1/(€*s)

Πράγματι, βλέπουμε ότι αυτή τη φορά συμφέρει η cache μας να είναι direct-mapped αν και η διαφορά μεταξύ των δύο περιπτώσεων είναι μικρή. Αυτό όμως υποδεικνύει ότι και η πιο συμφέρουσα επιλογή για την cache των 16kB είναι η direct-mapped. Επιπλέον, παρατηρούμε ότι και οι δύο υλοποιήσεις, αν και γρηγορότερες χρονικά από αυτές των 4kB, υστερούν στο λόγο απόδοσης προς κόστος. Αυτό μας δείχνει ότι η cache των 16kB θα έχει ακόμη χαμηλότερο λόγο απόδοσης και επομένως θα μπορούσαμε να την παραλείψουμε, ωστόσο δοκιμάσαμε την παρακάτω υλοποίηση προς απόδειξη των λεγομένων μας.

- Number of sets = 128, Block size: = 32, Set associativity = 1

Cost	Clock	Instructions	Total Cycles	CPI	Time	Value
50€	500MHz	3.424.091	3.715.859	1,085	0,00743 s	2,69116777574 * 1/(€*s)

Από το παραπάνω, διαπιστώνουμε ότι η επιλογή των 16kB data cache συμφέρει ακόμη λιγότερο, παρά το γεγονός ότι είναι η πιο γρήγορη χρονικά από όλες.

<u>Τελικά για τον επεξεργαστή rabbit επιλέγουμε:</u>

L1 program cache => Number of sets = 64, Block size = 32, Set Associativity = 1

L1 data cache => Number of sets = 16, Block size = 32, Set Associativity = 2, Write-back, Write allocate, LRU

3) Επεξεργαστής Puma:

Μετά από τη μελέτη του επεξεργαστή rabbit, μπορούμε να υποθέσουμε ότι το κόστος των μεγαλύτερων level 2 caches θα τις καθιστά μη συμφέρουσες, με αποτέλεσμα να επιλέξουμε τη μικρότερη. Ένας ακόμη παράγοντας που μας οδηγεί σε αυτή την υπόθεση είναι ότι το CPI πλησιάζει πάρα πολύ το ιδανικό με τη χρήση του δεύτερου επεξεργαστή και έτσι δεν υπάρχουν μεγάλα περιθώρια βελτίωσης που να μπορεί να προσφέρει η level 2 cache με αποτέλεσμα να μην είναι πιθανότατα η πιο συμφέρουσα από τις 3 εκδοχές επεξεργαστών. Επειδή όμως δεν ξέρουμε κατά πόσο επιταχύνουν ακριβώς το πρόγραμμα θα εξετάσουμε τις εξής περιπτώσεις:

- Number of sets = 64, Block size: = 32, Set associativity = 2

Cost	Clock	Instructions	Total Cycles	CPI	Time	Value
90€	490MHz	3.424.129	3.702.442	1,081	0,00756 s	1,47050094085 * 1/(€*s)

- Number of sets = 128, Block size: = 32, Set associativity = 2

Cost	Clock	Instructions	Total Cycles	CPI	Time	Value
115€	490MHz	3.424.129	3.683.402	1,076	0,00752 s	1,15677560179 * 1/(€*s)

- Number of sets = 256, Block size: = 32, Set associativity = 2

Cost	Clock	Instructions	Total Cycles	CPI	Time	Value
140€	490MHz	3.424.129	3.674.042	1,073	0,00750 s	0,952629284041 * 1/(€*s)

Σε όλες τις παραπάνω δοκιμές η level 1 program & data cache έχει τις ρυθμίσεις του επεξεργαστή rabbit.

Παρατηρούμε ότι όλες οι υλοποιήσεις επιταχύνουν την εκτέλεση του προγράμματος και βελτιώνουν το CPI, ωστόσο όλες έχουν σημαντικά χαμηλότερο λόγο απόδοσης προς κόστος. Εδώ να σημειώσουμε ότι αν μπορούσαμε να επιλέξουμε διαφορετική συσχετιστικότητα από αυτή του rabbit, η direct-mapped level 2 cache θα είχε καλύτερα αποτελέσματα, αφού δεν έχει πολλά capacity-misses και έτσι η μείωση που επιβάλλεται στο ρολόι επιβαρύνει τον επεξεργαστή περισσότερο από ότι τον βοηθά. Ακόμη και έτσι όμως, τα αποτελέσματα μας δείχνουν ότι η διαφορά μεταξύ των δύο είναι αρκετά μεγάλη ώστε να ανατραπεί η κατάταξή τους με βάση την απόδοση προς κόστος.

Τελικά για τον επεξεργαστή puma επιλέγουμε:

L1 program cache => Number of sets = 64, Block size = 32, Set Associativity = 1

L1 data cache => Number of sets = 16, Block size = 32, Set Associativity = 2, Write-back, Write allocate, LRU

L2 program & data cache => Number of sets = 64, Block size = 32, Set Associativity = 2, Write-back, Write allocate, LRU

Τελικά configurations:

Χαρακτηριστικό	Πρώτος επεξεργαστής (*turtle*)	Δεύτερος επεξεργαστής (*rabbit*)	Τρίτος επεξεργαστής (*puma*)
L1 program cache (μέγεθος, μέγεθος μπλοκ, συσχετιστικότητα, πολιτική αντικατάστασης)	-	8kB, 32 words, 1-way, -	8kB, 32 words, 1-way, -
L1 data cache (μέγεθος, μέγεθος μπλοκ, συσχετιστικότητα, πολιτική αντικατάστασης)	-	4kB, 32 words, 2-way, LRU	4kB, 32 words, 2-way, LRU
L2 cache (μέγεθος, μέγεθος μπλοκ, συσχετιστικότητα, πολιτική αντικατάστασης)	-	-	16kB, 32 words, 2-way, LRU
Κόστος	20€	40€	90€
Ρυθμός ρολογιού	500MHz	490MHz	490MHz
Κύκλοι εκτέλεσης	179.333.872	3.728.972	3.702.442
Χρόνος εκτέλεσης	0,35867 s	0,00761 s	0,00756 s
Λόγος απόδοσης/κόστους	0,1394047857283 * 1/(€*s)	3,285087686365 * 1/(€*s)	1,470500940850 * 1/(€*s)
Σύστημα για το οποίο είναι κατάλληλος (~150 λέξεις)	Οικιακές συσκευές (πχ. Κλιματιστικά)	Συσκευές χαμηλής υπολογιστικής ισχύος (πχ. Αμάξια)	Υπολογιστές desktop

Συμπεράσματα:

- Καλύτερος λόγος απόδοσης προς κόστος: **Rabbit**
- Καλύτερη ταχύτητα: **Puma**

Σχετικά με την καταλληλόλητα κάθε επεξεργαστή, σαν γενικό σχόλιο, όσο γρηγορότερος είναι ένας επεξεργαστής, τόσο περισσότερο επωφελείται από την ύπαρξη μεγάλης cache και πολλαπλών επιπέδων cache. Αν ο επεξεργαστής μας δεν είχε συχνότητα 500MHz, αλλά ήταν της τάξης των GHz όπως οι σημερινοί desktop computers, πιθανότατα να είχαμε πολύ διαφορετικά αποτελέσματα στην υλοποίηση puma προς όφελός της. Από την άλλη βασικό ρόλο για την επιλογή της συχνότητας παίζει και το που θέλουμε να χρησιμοποιήσουμε τον επεξεργαστή. Αν για παράδειγμα έχουμε ως βασικό στόχο τη μεγιστοποίηση της αυτονομίας ενός κόμβου IoT που έχει τροφοδοσία μπαταρίας, τότε σίγουρα επιλέγουμε κάποιον επεξεργαστή με χαμηλή συχνότητα ρολογιού και ως αποτέλεσμα το να τοποθετήσουμε σε αυτόν cache δεν θα τον επιταχύνει, αφού τα δεδομένα θα φτάνουν γρηγορότερα από ότι μπορεί να τα επεξεργαστεί.

Συμπληρωματικό υλικό:

Ενδεικτικά αποτελέσματα του πίνακα μετά από ταξινόμηση με quicksort. Απεικόνιση δεδομένων σε bytes. Αναζήτηση του αριθμού 92 με binary search.

Memory

Byte — array start — Direct — POS

Address	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+10	+11	+12	+13	+14	+15	+16	+17	+18	+19	+20	+21	+22	+23	+24	+25	+26	+27	+28	+29	+30
0x80020115	06	FF	FF	08	00	80	39	01	24	48	22	21	29	00	01	AE	E9	00	00	03	E0	00	08	03	E0	00	08	00	00	70	4E
0x80020134	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
0x80020153	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
0x80020172	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
0x80020191	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
0x800201B0	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
0x800201CF	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
0x800201EE	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
0x8002020D	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
0x8002022C	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
0x8002024B	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	02
0x8002026A	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02
0x80020289	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02
0x800202A8	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02
0x800202C7	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02
0x800202E6	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02
0x80020305	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02
0x80020324	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02
0x80020343	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02
0x80020362	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02
0x80020381	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02
0x800203A0	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	03	03	03	03	03	03	03	03	03	03
0x800203BF	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03
0x800203DE	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03
0x800203FD	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03
0x8002041C	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03
0x8002043B	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03
0x8002045A	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03
0x80020479	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03

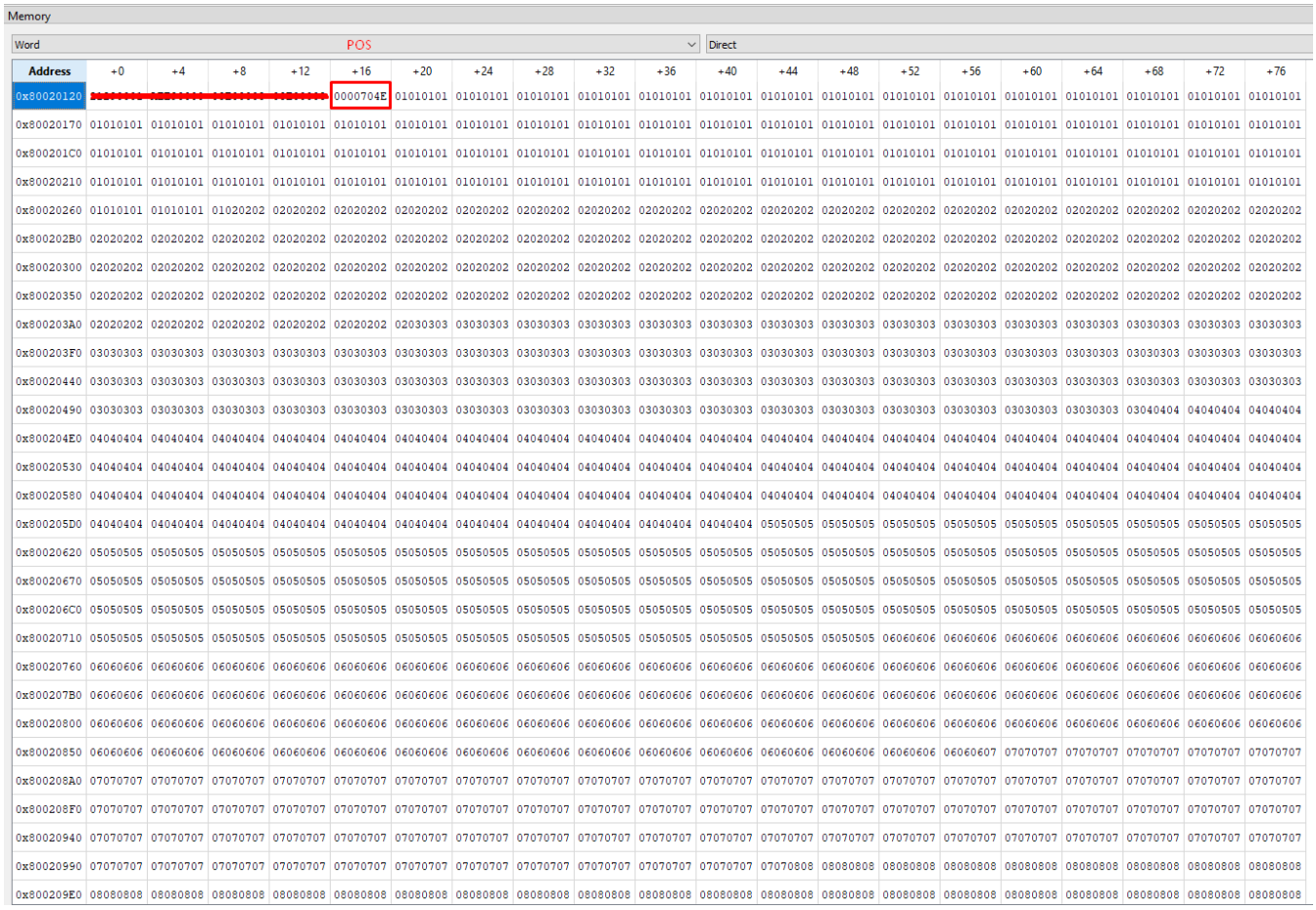
Memory

Byte — Direct

Address	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+10	+11	+12	+13	+14	+15	+16	+17	+18	+19	+20	+21	+22	+23	+24	+25	+26	+27	+28	+29	+30
0x80029A4F	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D
0x80029A6E	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D
0x80029A8D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D
0x80029AAC	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D
0x80029ACB	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D
0x80029AEA	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7D	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E
0x80029B09	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E
0x80029B28	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E
0x80029B47	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E
0x80029B66	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E
0x80029B85	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E
0x80029BA4	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E
0x80029BC3	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E
0x80029BE2	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E
0x80029C01	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E
0x80029C20	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E	7E
0x80029C3F	7E	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F
0x80029C5E	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F
0x80029C7D	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F
0x80029C9C	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F
0x80029CBB	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F
0x80029CDA	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F
0x80029CF9	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F
0x80029D18	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F
0x80029D37	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F
0x80029D56	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F
0x80029D75	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

array end

Απεικόνιση των ίδιων αποτελεσμάτων ως words. Σε κάθε θέση υπάρχουν 4 αριθμοί του 1 byte.



Memory

Word | POS | Direct

Address	+0	+4	+8	+12	+16	+20	+24	+28	+32	+36	+40	+44	+48	+52	+56	+60	+64	+68	+72	+76
0x80020120	[redacted]				0000704E	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101
0x80020170	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101
0x800201C0	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101
0x80020210	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101
0x80020260	01010101	01010101	01020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202
0x800202B0	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202
0x80020300	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202
0x80020350	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202	02020202
0x800203A0	02020202	02020202	02020202	02020202	02020202	02030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303
0x800203F0	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303
0x80020440	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303
0x80020490	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03030303	03040404	04040404	04040404
0x800204E0	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404
0x80020530	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404
0x80020580	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404
0x800205D0	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	04040404	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505
0x80020620	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505
0x80020670	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505
0x800206C0	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505
0x80020710	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	05050505	06060606	06060606	06060606	06060606	06060606	06060606	06060606
0x80020760	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606
0x800207B0	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606
0x80020800	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606
0x80020850	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060606	06060607	07070707	07070707	07070707	07070707	07070707
0x800208A0	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707
0x800208F0	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707
0x80020940	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707
0x80020990	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070707	07070808	08080808	08080808	08080808	08080808	08080808	08080808	08080808	08080808
0x800209E0	08080808	08080808	08080808	08080808	08080808	08080808	08080808	08080808	08080808	08080808	08080808	08080808	08080808	08080808	08080808	08080808	08080808	08080808	08080808	08080808

Ευχαριστούμε για το χρόνο σας.