Ενότητα 2η: *Η απόδοση της ΚΜΕ*

Σκοπός **Ο σκοπός της ενότητας αυτής είναι να παρουσιάσει τις βασικές αρχές σχεδίασης των υπολογιστικών συστημάτων.**

Προσδοκώμενα Αποτελέσματα **Όταν θα έχετε μελετήσει την ενότητα, θα είστε σε θέση να:**

 **προσδιορίζετε τις ποσοτικές αρχές σχεδιασμού των υπολογιστών,**

 **αναφέρετε το νόμο του Amdahl και τη σημασία του,**

 **προσδιορίζετε και να συγκρίνετε την απόδοση των ΚΜΕ δύο υπολογιστών, χρησιμοποιώντας την εξίσωση της απόδοσης της ΚΜΕ,**

 **εξηγείτε τον τρόπο με τον οποίο μετρώνται οι συνιστώσες της απόδοσης της ΚΜΕ,**

 **αναφέρετε τους δείκτες της απόδοσης της ΚΜΕ και να επισημαίνετε τις όποιες επιτυχίες – αποτυχίες,**

 **αναγνωρίζετε τη σχέση ανάμεσα στην τοπικότητα της αναφοράς και την απόδοση,**

 **εξάγετε συμπεράσματα για τον ρυθμό επιτυχίας.**

χρόνος υπολογισμού, βελτιωμένο κλάσμα, χτύποι ρολογιού, κύκλος ρολογιού, χρόνος ΚΜΕ, τοπικότητα αναφοράς, τοπικότητα χώρου, τοπικότητα χρόνου, κύκλοι καθυστέρησης μνήμης, μήκος μονοπατιού εντολών

Ποσοτικές Αρχές Σχεδιασμού Υπολογιστών

Έχοντας ήδη ορίσει, υπολογίσει και συνοψίσει την **απόδοση** μπορούμε να διερευνήσουμε μερικές κατευθυντήριες γραμμές και αρχές που είναι χρήσιμες για τον σχεδιασμό και την ανάλυση των υπολογιστών. Συγκεκριμένα θα εισάγουμε μερικές σημαντικές παρατηρήσεις για το πώς σχεδιάζονται οι υπολογιστές, στηριζόμενοι στην απόδοση και τον λόγο κόστος /απόδοση. Στη συνέχεια, θα εισάγουμε δύο εξισώσεις που χρησιμοποιούνται για την αξιολόγηση εναλλακτικών σχεδίων.

Κάνοντας την μέση περίπτωση πιο γρήγορη

Η πιο σημαντική και γενική αρχή στο σχεδιασμό υπολογιστών είναι να κάνουμε την μέση περίπτωση πιο γρήγορη. Αυτό μπορεί να επιτευχθεί δημιουργώντας ένα σχέδιο και μεροληπτώντας υπέρ της πιο συχνής περίπτωσης εις βάρος της πιο σπάνιας. Η ίδια αρχή εφαρμόζεται και όταν πρέπει να αποφασίσουμε για την διάθεση των πόρων, αφού η πίεση για να επιταχύνουμε ένα περιστατικό είναι μεγαλύτερη όταν αυτό εμφανίζεται συχνά. Εάν βελτιώσουμε ένα πρόγραμμα με μεγάλη συχνότητα εμφάνισης είναι εμφανές ότι θα βελτιώσουμε και την απόδοση.
Εφαρμόζοντας αυτή την απλή αρχή, προσδιορίζουμε την πιο συχνή περίπτωση και βρίσκουμε κατά πόσο θα βελτιωθεί η απόδοση κάνοντας αυτή την περίπτωση πιο γρήγορη. Ένας πρωταρχικός νόμος που χρησιμοποιείται για τον ποσοτικό προσδιορισμό αυτής της αρχής είναι ο νόμος του Amdahl.

Ο νόμος του Amdahl.

Ο νόμος του Amdahl χρησιμοποιείται για τον υπολογισμό του κέρδους στην απόδοση, όταν βελτιωθεί κάποιο τμήμα (portion) του υπολογιστή. Πιο συγκεκριμένα, η βελτίωση της απόδοσης που μπορούμε να επιτύχουμε χρησιμοποιώντας έναν πιο γρήγορο τρόπο εκτέλεσης, περιορίζεται από το κλάσμα του χρόνου που μπορεί να χρησιμοποιηθεί αυτός ο τρόπος εκτέλεσης.

Ο νόμος του Amdahl δίνει τον ορισμό της **επιτάχυνσης** που μπορούμε να επιτύχουμε χρησιμοποιώντας ένα συγκεκριμένο χαρακτηριστικό. Εάν υποθέσουμε ότι μπορούμε να κάνουμε κάποια **βελτίωση** στον υπολογιστή μας η οποία επηρεάζει την απόδοση όταν χρησιμοποιείται , τότε η επιτάχυνση ορίζετε ως εξής (τύπος 1.13):

$$\textbf{επιτάχυνση} = \frac{\textbf{Απόδοση ολόκληρης της εργασίας χρησιμοποιώντας την βελτίωση όταν είναι δυνατό}}{\textbf{Απόδοση ολόκληρης της εργασίας χωρίς την χρήση της βελτίωση}} \quad \textbf{(Τύπος 1.13)}$$

ή εναλλακτικά (τύπος 1.14):

$$\textbf{επιτάχυνση} = \frac{\textbf{χρόνος εκτέλεσης ολόκληρης της εργασίας χωρίς την χρήση της βελτίωσης}}{\textbf{χρόνος εκτέλεσης ολόκληρης της εργασίας χρησιμοποιώντας την βελτίωση όταν είναι δυνατό}} \quad \textbf{(Τύπος 1.14)}$$

Η επιτάχυνση δηλώνει πόσο επισπεύδεται η ολοκλήρωση μιας εργασίας, αν αυτή εκτελεστεί σε έναν υπολογιστή που έχει υποστεί κάποια βελτίωση συγκριτικά με τον αρχικό υπολογιστή.

Ο νόμος του Amdahl μας δίνει έναν γρήγορο τρόπο να υπολογίζουμε την επιτάχυνση που προκύπτει από κάποια βελτίωση που κάνουμε στο σύστημά μας, ο οποίος εξαρτάται από δύο παράγοντες:

1. *Από το κλάσμα του χρόνου υπολογισμού στον αρχικό υπολογιστή που μπορεί να μετατραπεί λόγω της βελτίωσης*.

Παράδειγμα

Εάν ο συνολικός χρόνος εκτέλεσης ενός προγράμματος είναι 60 sec και τα 20 sec από αυτά μπορούν να χρησιμοποιήσουν την εξέλιξη, τότε το κλάσμα είναι 20/60. Αυτή η τιμή η οποία ονομάζεται Κλάσμα$_{\text{βελτιωμένο}}$ (Fraction$_{\text{enhanced}}$) είναι πάντα μικρότερο ή ίσο της μονάδας.

$$Κλάσμα_{βελτιωμένο} \leq 1$$

2. *Η βελτίωση που πετυχαίνουμε με τον βελτιωμένο τρόπο εκτέλεσης, δηλαδή πόσο πιο γρήγορα θα εκτελεστεί η εργασία αν χρησιμοποιηθεί ο βελτιωμένος τρόπος για ολόκληρο το πρόγραμμα.* Αυτή η τιμή είναι ο χρόνος του αρχικού τρόπου εκτέλεσης προς το χρόνο του βελτιωμένου τρόπου.

Παράδειγμα

Εάν ο βελτιωμένος τρόπος χρειάζεται 2sec για ένα τμήμα του προγράμματος το οποίο μπορεί να χρησιμοποιήσει ολόκληρο τον τρόπο αυτό, ενώ ο αρχικός τρόπος χρειάζεται 5 sec για το ίδιο τμήμα, η βελτίωση είναι 5/2.

Θα ονομάζουμε αυτή την τιμή, η οποία είναι **πάντα μεγαλύτερη της μονάδας**, **επιτάχυνση**$_{\text{βελτιωμένη}}$

Εάν χρησιμοποιήσουμε τον αρχικό αλλά βελτιωμένο υπολογιστή, ο χρόνος εκτέλεσης θα είναι το άθροισμα του χρόνου που καταναλώνεται χρησιμοποιώντας τμήμα του υπολογιστή που δεν έχει βελτιωθεί και του χρόνου που καταναλώθηκε χρησιμοποιώντας την βελτίωση (τύπος 1.15 - 1.16):

$$\text{νέος χρόνος εκτέλεσης} = \text{παλιός χρόνος εκτέλεσης} \times \left((1 - \text{Κλάσμα}_{\text{βελτιωμένο}}) + \frac{\text{Κλάσμα}_{\text{βελτιωμένο}}}{\text{επιτάχυνση}_{\text{βελτιωμένη}}} \right) \quad \textbf{(Τύπος 1.15)}$$

Η ολική επιτάχυνση (overall speedup) είναι ο λόγος των χρόνων εκτέλεσης:

$$\text{ολική επιτάχυνση} = \frac{\text{παλιός χρόνος εκτέλεσης}}{\text{νέος χρόνος εκτέλεσης}} = \frac{1}{(1 - \text{κλάσμα}_{\text{βελτιωμένο}}) + \frac{\text{Κλάσμα}_{\text{βελτιωμένο}}}{\text{επιτάχυνση}_{\text{βελτιωμένη}}}} \quad \textbf{(Τύπος 1.16)}$$

Όσο προσθέτουμε βελτιώσεις η αυξανόμενη βελτίωση στην επιτάχυνση που επιτυγχάνουμε από μία επιπρόσθετη βελτίωση, ελαττώνεται. Μία απόρροια του νόμου του Amdahl είναι ότι αν μία βελτίωση χρησιμοποιείται μόνο σε ένα κλάσμα της εργασίας, δεν μπορούμε να επιταχύνουμε την εργασία αυτή περισσότερο από το αντίστροφο του 1 μείον αυτό το κλάσμα..

Μπορούμε να χρησιμοποιήσουμε τον νόμο του Amdahl για να βρούμε πόσο θα συμβάλλει στην βελτίωση της απόδοσης η βελτίωση που θα κάνουμε στον υπολογιστή και πως θα κατανεμηθούν οι πόροι του συστήματος για την βελτίωση του λόγου

κόστος/απόδοση. Ακόμη μπορούμε να τον χρησιμοποιήσουμε για να συγκρίνουμε δύο εναλλακτικά σχέδια υπολογιστή.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 1

Έστω ότι ο βελτιωμένος υπολογιστής είναι 10 φορές ταχύτερος από τον αρχικό, αλλά χρησιμοποιεί μόνον το 40% του χρόνου του. Να βρείτε ποια είναι η ολική επιτάχυνση που πραγματοποιήθηκε από την ενσωμάτωση της βελτίωσης;

ΑΠΑΝΤΗΣΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ 1

Κλάσμα $_{\text{βελτιωμένο}}$ = 0.4

Επιτάχυνση $_{\text{βελτιωμένη}}$ = 10

$$\textbf{Επιτάχυνση}_{\text{Ολική}} = \frac{1}{0.6 + (0.4/10)} = 1/0.604 \approx 1.66$$

Παράδειγμα

Οι υλοποιήσεις της τετραγωνικής ρίζας κινητής υποδιαστολής (FPSQR) διαφέρουν σημαντικά στην απόδοση. Υποθέτουμε ότι η τετραγωνική ρίζα FP κινητής υποδιαστολής είναι υπεύθυνη για το 20% του χρόνου εκτέλεσης ενός σημαντικού προγράμματος δοκιμής σε έναν υπολογιστή. Μία πιθανή πρόταση είναι να προστεθεί το κατάλληλο υλικό για τις FPSQR το οποίο θα επιταχύνει αυτή την λειτουργία κατά έναν παράγοντα ίσο με 10. Η άλλη εναλλακτική πρόταση είναι να εκτελούνται όλες οι εντολές FP πιο γρήγορα. Οι FP εντολές αποτελούν το 50% του συνολικού χρόνου εκτέλεσης. Τα μέλη της ομάδας σχεδιασμού υπολογιστών πιστεύουν ότι μπορούν να βελτιώσουν την εκτέλεση των εντολών FP ώστε να εκτελούνται 2 φορές πιο γρήγορα καταβάλλοντας την ίδια προσπάθεια που απαιτείται και για να επιταχύνουν την εκτέλεση της FPSQR. Συγκρίνετε τις δύο αυτές εναλλακτικές προτάσεις σχεδίων.

Απάντηση:

Μπορούμε να συγκρίνουμε τις δύο αυτές εναλλακτικές λύσεις συγκρίνοντας τις επιταχύνσεις χρησιμοποιούμε τον τύπο (1.16) :

$$\text{επιτάχυνση}_{FPSQR} = \frac{1}{(1-0.2)+\frac{0.4}{10}} = \frac{1}{0.802} = 1.25$$

$$\text{επιτάχυνση}_{FP} = \frac{1}{(1-0.5)+\frac{0.5}{2.0}} = \frac{1}{0.75} = 1.33$$

Βελτιώνοντας την απόδοση των λειτουργιών FP (κινητής υποδιαστολής) η συνολική βελτίωση είναι καλύτερη εξαιτίας της μεγαλύτερης συχνότητας εκτέλεσης αυτών των εντολών.

 Στο προηγούμενο παράδειγμα έπρεπε να γνωρίζουμε τον χρόνο που καταναλώνεται από τις βελτιωμένες λειτουργίες κινητής υποδιαστολής. Συνήθως είναι δύσκολο να υπολογίσουμε αυτούς τους χρόνους απ' ευθείας. Ακολουθεί ένας άλλος τρόπος σύγκρισης των εναλλακτικών σχεδίων, που χρησιμοποιεί μια εξίσωση ανάλυσης του χρόνου εκτέλεσης της ΚΜΕ σε τρία συστατικά μέρη. Εάν γνωρίζουμε τον τρόπο επιρροής της εναλλακτικής λύσης στα συστατικά αυτά, προσδιορίζεται και η επιρροή στη συνολική απόδοση. Επιπλέον μπορούμε να φτιάξουμε εξομοιωτές που υπολογίζουν αυτά τα συστατικά της εξίσωσης πριν σχεδιαστεί το υλικό (hardware) του υπολογιστή.

Η εξίσωση της απόδοσης της ΚΜΕ

Οι περισσότεροι υπολογιστές έχουν κατασκευαστεί ώστε να χρησιμοποιούν ένα ρολόι που λειτουργεί με σταθερό ρυθμό. Τα γεγονότα που συμβαίνουν σε διακριτό χρόνο καλούνται **χτύποι ρολογιού, περίοδοι ρολογιού, κύκλοι ρολογιού, ρολόγια, ή χτύποι .**
Οι σχεδιαστές υπολογιστών αναφέρονται στον χρόνο μίας περιόδου ρολογιού είτε με την διάρκειά της π.χ. 2 ns είτε με την συχνότητά του π.χ. 500 MHz.
Ο **χρόνος της ΚΜΕ** ενός προγράμματος μπορεί να εκφραστεί με δύο τρόπους:

Χρόνος ΚΜΕ = αριθμός κύκλων ρολογιού ΚΜΕ για ένα πρόγραμμα *διάρκεια κύκλου ρολογιού **(Τύπος 1.17)**

ή

$$\textbf{Χρόνος ΚΜΕ} = \frac{\textbf{αριθμός κύκλων ρολογιού ΚΜΕ για ένα πρόγραμμα}}{\textbf{Συχνότητα ρολογιού}} \quad \textbf{(Τύπος 1.18)}$$

Εκτός από τον αριθμό των κύκλων ρολογιού που απαιτούνται για την εκτέλεση ενός προγράμματος μπορούμε να υπολογίσουμε το **πλήθος των εντολών που εκτελούνται (IC)** ή το **μήκος μονοπατιού εντολών.** Εάν γνωρίζουμε το πλήθος των κύκλων ρολογιού και τον αριθμό των εντολών (instruction count) μπορούμε να υπολογίσουμε το **μέσο πλήθος των κύκλων ρολογιού ανά εντολή (CPI):**

$$\textbf{Κύκλοι ρολογιού ανά εντολή (CPI)} = \frac{\textbf{Αριθμός κύκλων ρολογιού ΚΜΕ για ένα πρόγραμμα}}{\textbf{πλήθος εντολών που εκτελούνται}} \quad \textbf{(Τύπος 1.19)}$$

Αντικαθιστώντας το πλήθος εντολών που εκτελούνται από την παραπάνω σχέση στον τύπο 17 προκύπτει:

Χρόνος ΚΜΕ = πλήθος εντολών που εκτελούνται * κύκλοι ρολογιού ανά εντολή * διάρκεια κύκλου ρολογιού **(Τύπος 1.20)**

ή

$$\textbf{Χρόνος ΚΜΕ} = \frac{\textbf{πλήθος εντολών που εκτελούνται * κύκλοι ρολογιού ανά εντολή}}{\textbf{συχνότητα ρολογιού}}$$ **(Τύπος 1.21)**

Επεκτείνοντας την πρώτη από τις δύο παραπάνω σχέσεις στις μονάδες μέτρησης προκύπτει:

$$\frac{\textbf{εντολές}}{\textbf{πρόγραμμα}} \times \frac{\textbf{κύκλοι ρολογιού}}{\textbf{εντολή}} \times \frac{\textbf{δευτερόλεπτα}}{\textbf{κύκλος ρολογιού}} = \frac{\textbf{δευτερόλεπτα}}{\textbf{κύκλος ρολογιού}} = \textbf{χρόνος ΚΜΕ}$$

Όπως δείχνει ο τύπος (1.20) η απόδοση της ΚΜΕ εξαρτάται από 3 χαρακτηριστικά από:

1.τον κύκλο ή συχνότητα ρολογιού
2. τον αριθμό κύκλων ανά εντολή
3.το πλήθος εντολών

Επιπλέον ο **χρόνος ΚΜΕ** εξαρτάται εξίσου και από τα 3 αυτά χαρακτηριστικά: Μία 10% βελτίωση σε κάποιο από αυτά οδηγεί σε 10% βελτίωση του ΚΜΕ χρόνου. Δυστυχώς είναι δύσκολο να αλλάξουμε μία από τις παραμέτρους αυτές χωρίς να επηρεαστούν και οι άλλες γιατί οι βασικές τεχνολογίες που σχετίζονται με την αλλαγή κάθε χαρακτηριστικού αλληλεξαρτώνται.

- ***Διάρκεια κύκλου ρολογιού*** - εξαρτάται από την τεχνολογία του υλικού και την οργάνωση
- ***Κύκλοι ρολογιού ανά εντολή (CPI)*** - εξαρτάται από την οργάνωση και το σύνολο εντολών της αρχιτεκτονικής
- ***Πλήθος εντολών(IC)*** - εξαρτάται από το σύνολο εντολών της αρχιτεκτονικής και από την τεχνολογία του μεταγλωττιστή.

Ευτυχώς πολλές τεχνικές βελτίωσης απόδοσης αρχικά βελτιώνουν ένα συστατικό της απόδοσης της ΚΜΕ με μικρή ή προβλέψιμη επιρροή στα άλλα δύο. Μερικές φορές είναι

χρήσιμο καθώς σχεδιάζουμε την ΚΜΕ να υπολογίζουμε τον **ολικό αριθμό κύκλων ρολογιού της ΚΜΕ**

$$\textbf{ΚΜΕ}_{\text{ κύκλοι ρολογιού}} = \sum_{i=1}^{n} \textbf{CPI}_i \times \textbf{IC}_i \qquad \textbf{(Τύπος 1.22)}$$

Όπου **CPIi** : *μέσος αριθμός κύκλων ρολογιού για την εντολή i*
ICi : *πόσες φορές εκτελείται η i εντολή σε ένα πρόγραμμα*
Ετσι μπορούμε να εκφράσουμε τον χρόνο της ΚΜΕ ως εξής:

$$\textbf{χρόνος ΚΜΕ} = \left(\sum_{i=1}^{n} \textbf{CPI}_i \times \textbf{IC}_i \right) \times \textbf{κύκλος ρολογιού} \qquad \textbf{(Τύπος 1.23)}$$

Τέλος το **ολικό CPI** (κύκλοι ρολογιού ανά εντολή) είναι :

$$\textbf{CPI} = \frac{\sum_{i=1}^{n} \textbf{CPI}_i \times \textbf{IC}_i}{\textbf{IC}} = \sum_{i=1}^{n} \textbf{CPI}_i \times \left(\frac{\textbf{IC}_i}{\textbf{IC}} \right) \qquad \textbf{(Τύπος 1.24)}$$

Όπου **IC**: (instruction count) πλήθος εντολών που εκτελούνται
IC_i: πόσες φορές εκτελείται η εντολή i σε ένα πρόγραμμα

Παρακάτω θεωρούμε την τροποποίηση του προηγούμενου παραδείγματος (για το νόμο του Amdahl) για να χρησιμοποιήσουμε τις μετρήσεις της συχνότητας των εντολών και τις τιμές των κύκλων ρολογιού της κάθε εντολής το οποίο είναι πιο εύκολο να βρούμε.

Παράδειγμα

Έστω ότι έχουμε τις παρακάτω μετρήσεις:

- **Συχνότητα των λειτουργιών κιν.υποδιαστολής (FP) = 25%**
- **Μέσος CPI των λειτουργιών κιν.υποδιαστολής (FP) = 4**
- **Μέσος CPI των υπολοίπων εντολών = 1,33**
- **Συχνότητα των λειτουργιών FPSPQ = 2%**
- **CPI των FPSQR =20**

(FPSQR=Floating Point Square Root δηλ τετραγωνική ρίζα κινητής υποδιαστολής) Ας υποθέσουμε ότι η πρώτη εναλλακτική λύση είναι να μειώσουμε το CPI των FPSQR σε 2 και η δεύτερη να μειώσουμε το μέσο CPI όλων των λειτουργιών κινητής υποδιαστολής χρησιμοποιώντας την εξίσωση απόδοσης της ΚΜΕ. Να συγκρίνετε αυτές τις δύο εναλλακτικές προτάσεις χρησιμοποιώντας την εξίσωση απόδοσης της ΚΜΕ.

Απάντηση:

Καταρχήν παρατηρούμε ότι μόνο οι κύκλοι ρολογιού ανά εντολή (CPI) αλλάζουν. Η συχνότητα ρολογιού (clock rate) και ο αριθμός εντολών (IC) παραμένουν τα ίδια. Ετσι υπολογίζουμε τους αρχικούς κύκλους ρολογιού ανά εντολή ($CPI_{αρχικό}$) χωρίς την βελτίωση.

$$\mathbf{CPI_{αρχικό}} = \sum_{i=1}^{n} \mathbf{CPI_i} \times \left(\frac{\mathbf{IC_i}}{\mathbf{IC}} \right) = \mathbf{(4 \times 25\%) + (1{,}33 \times 75\%) = 2.0}$$

Υπολογίζουμε στην συνέχεια τους κύκλους ρολογιού ανά εντολή για την βελτιωμένη FPSQR:

$CPI_{βελτιωμένου\ FPSQR} = CPI_{αρχικό} - 2\% * (CPI_{παλιού\ FPSQR} - CPI_{βελτιωμένου\ FPSQR})$
$= 2.0 - 2\% * (20-2) = 1{,}64$

Καθώς και τους κύκλους ρολογιού ανά εντολή για την βελτίωση όλων των λειτουργιών κινητής υποδιαστολής:

$CPI_{βελτιωμένου\ FP} = (75\% * 1{,}33) + (25\% * 2{,}0) = 1{,}5$

Εφόσον οι κύκλοι ρολογιού ανά εντολή μετά την βελτίωση όλων των λειτουργιών κινητής υποδιαστολής είναι λιγότεροι από αυτούς που βρίσκουμε μετά την βελτίωση της τετραγωνικής ρίζας κινητής υποδιαστολής, η απόδοση της πρώτης λύσης είναι καλύτερη. Συγκεκριμένα η επιτάχυνση που πετυχαίνουμε με την βελτίωση όλων των λειτουργιών FP είναι:

$$\mathbf{επιτάχυνση_{βελτιωμένου\ FP}} = \frac{\mathbf{αρχικός\ χρόνος\ ΚΜΕ}}{\mathbf{χρόνος\ ΚΜΕ_{βελτιωμένουFP}}} = \frac{\mathbf{IC \times κύκλοι\ ρολογιού \times CPI_{αρχικό}}}{\mathbf{IC \times κύκλοι\ ρολογιού \times CPI_{βελτιωμένουFP}}} =$$

$$\frac{\mathbf{CPI_{αρχικό}}}{\mathbf{CPI_{βελτιωμένουFP}}} = \frac{\mathbf{2{,}00}}{\mathbf{1{,}5}} = \mathbf{1{,}33}$$

Παρατηρούμε ότι η επιτάχυνση αυτή είναι ίδια με εκείνη που βρήκαμε και με την βοήθεια του νόμου του Amdahl.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 2

Μπορείτε να εξηγήσετε το λόγο για τον οποίο η εξίσωση της ΚΜΕ πλεονεκτεί έναντι του νόμου του Amdahl;

ΑΠΑΝΤΗΣΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ 2

Η εξίσωση απόδοσης της ΚΜΕ πλεονεκτεί έναντι του νόμου του Amdahl γιατί μπορούμε να μετρήσουμε τα συστατικά της μέρη, ενώ είναι δύσκολο να μετρήσουμε το κλάσμα του χρόνου εκτέλεσης για ένα ορισμένο σύνολο εντολών.

Στην πράξη, αθροίζουμε το γινόμενο του πλήθους των εντολών που εκτελούνται με τους κύκλους ρολογιού ανά εντολή για καθεμία από τις εντολές του συνόλου.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 3

Έστω ότι έχουμε τις ακόλουθες εναλλακτικές λύσεις για τις εντολές διακλάδωσης με συνθήκη:
$ΚΜΕ_A$: ένας κώδικας συνθήκης ορίζεται από εντολή σύγκρισης και ακολουθείται από μία διακλάδωση η οποία ελέγχει τον κώδικα συνθήκης.
$ΚΜΕ_B$: στην διακλάδωση περιέχεται μία σύγκριση.
Και στις δύο ΚΜΕ, η εντολή διακλάδωσης με συνθήκη διαρκεί δύο κύκλους ρολογιού και όλες οι άλλες εντολές διαρκούν ένα κύκλο ρολογιού. Στην πρώτη ΚΜΕ το 20% των εντολών που εκτελούνται είναι συγκρίσεις. Επειδή η πρώτη ΚΜΕ δεν περιέχει τη σύγκριση μέσα στη διακλάδωση, υποθέτουμε ότι η διάρκεια του κύκλου ρολογιού της είναι 1.25 φορές πιο γρήγορη από αυτή της δεύτερης ΚΜΕ.
α) Να βρείτε ποια από τις δύο ΚΜΕ είναι πιο γρήγορη.
β) Ποια από τις δύο ΚΜΕ είναι πιο γρήγορη στην περίπτωση που η διάρκεια του κύκλου ρολογιού της είναι 1.1 φορές πιο γρήγορη από αυτή της δεύτερης ΚΜΕ;

ΑΠΑΝΤΗΣΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ 3

α) Εφόσον έχουμε αγνοήσει τα θέματα που προκύπτουν από το σύστημα μπορούμε να χρησιμοποιήσουμε την εξίσωση απόδοσης της ΚΜΕ. Έτσι έχουμε ότι:

CPI_A=0,20*2+0,80*1=1,2

Αφού 20%των διακλαδώσεων χρειάζονται 2 κύκλους ρολογιού και οι υπόλοιπες εντολές χρειάζονται 1 κύκλο ρολογιού. Η απόδοση της $ΚΜΕ_A$ είναι:

Χρόνος $ΚΜΕ_A$= πλήθος εντολών *1,2 *κύκλος $ρολογιού_A$

Ακόμα ισχύει ότι κύκλος $ρολογιού_B$ = 1,25 * κύκλος $ρολογιού_A$ αφού η πρώτη ΚΜΕ έχει συχνότητα ρολογιού η οποία είναι 1,25 φορές μεγαλύτερη. Στην $ΚΜΕ_B$ δεν εκτελούνται συγκρίσεις έτσι 20% / 80% ή 25% των εντολών είναι τώρα διακλαδώσεις, και χρειάζονται 2 κύκλους ρολογιού ενώ το υπόλοιπο 75% των εντολών χρειάζονται από έναν κύκλο ρολογιού. 'Ετσι:

CPI_B=0,25*2+0,75*1=1,25

Επειδή η $ΚΜΕ_B$ δεν εκτελεί συγκρίσεις πλήθος $εντολών_B$= πλήθος $εντολών_A$ * 0.8
Επομένως η απόδοση της $ΚΜΕ_B$ είναι:

Χρόνος $ΚΜΕ_B$ = πλήθος $εντολών_B$ * CPI_B* κύκλος $ρολογιού_B$
=0,8* πλήθος $εντολών_A$* 1,25 *(1,25 *κύκλος $ρολογιού_A$)
=1,25 * πλήθος $εντολών_A$* κύκλος $ρολογιού_A$
Κάτω από αυτές τις υποθέσεις η $ΚΜΕ_A$, με πιο σύντομο κύκλο ρολογιού ,είναι πιο γρήγορη από την $ΚΜΕ_B$,η οποία εκτελεί λιγότερες εντολές.

β) Εάν η $ΚΜΕ_A$ ήταν μόνο 1,1 φορές πιο γρήγορη ,τότε:

κύκλος $ρολογιού_B$=1,10 * κύκλος $ρολογιού_A$
και η απόδοση της $ΚΜΕ_B$ είναι

Χρόνος $ΚΜΕ_B$ = πλήθος $εντολών_B$ * CPI_B* κύκλος $ρολογιού_B$
=0,8* πλήθος $εντολών_A$* 1,25 *(1,10 *κύκλος $ρολογιού_A$)
=1,10 * πλήθος $εντολών_A$* κύκλος $ρολογιού_A$
Με αυτή την βελτίωση στην $ΚΜΕ_B$,η οποία εκτελεί λιγότερες εντολές τώρα είναι πιο γρήγορη.

Μέτρηση των συνιστωσών της απόδοσης της ΚΜΕ

Για να προσδιορίσουμε την απόδοση χρησιμοποιώντας την εξίσωση απόδοσης της ΚΜΕ, χρειαζόμαστε μετρήσεις για κάθε μία από τις συνιστώσες της εξίσωσης. Αν εξαιρέσουμε τον κύκλο ρολογιού, τα άλλα δύο συστατικά της εξίσωσης της απόδοσης της ΚΜΕ μπορούν να μετρηθούν πιο εύκολα. Πιο συγκεκριμένα:

 **Κύκλος ρολογιού:** Για να προσδιορίσουμε τον κύκλο ρολογιού αρκεί να προσδιορίσουμε μόνο έναν αριθμό.
Αυτό είναι εύκολο για μία υπάρχουσα ΚΜΕ, αλλά η εκτίμηση του κύκλου ρολογιού ενός σχεδίου που βρίσκεται σε εξέλιξη είναι πολύ δύσκολο. Για την ανάλυση του κύκλου ρολογιού ενός ολοκληρωμένου σχεδίου χρησιμοποιούμε εργαλεία χαμηλού επιπέδου που ονομάζονται εκτιμητές συγχρονισμού. Είναι πολύ πιο δύσκολο να υπολογίσουμε τον κύκλο ρολογιού ενός σχεδίου που δεν έχει ακόμα ολοκληρωθεί ή για μια εναλλακτική πρόταση όπου δεν υπάρχει ούτε το σχέδιο. Στην πράξη, οι σχεδιαστές προσδιορίζουν έναν επιθυμητό κύκλο ρολογιού και εκτιμούν την επιρροή του κύκλου εξετάζοντας το μονοπάτι το οποίο πιστεύουν ότι είναι κρίσιμο κατά την σχεδίαση. Το δυσκολότερο κομμάτι κατά την σχεδίαση αποδεικνύεται ότι είναι τελικά η **μονάδα ελέγχου** και όχι η **δίοδος δεδομένων** ενός επεξεργαστή. Η μονάδα ελέγχου είναι το τελευταίο τμήμα που φτιάχνεται και το πιο δύσκολο είναι να υπολογιστεί ο συγχρονισμός του. Έτσι, οι σχεδιαστές βασιζόμενοι στις εκτιμήσεις τους και στην εμπειρία τους, προσπαθούν να επιτύχουν τον επιθυμητό κύκλο ρολογιού. Κατά συνέπεια, μερικές φορές αυξάνεται το πλήθος των κύκλων ρολογιού ορισμένων εντολών, εξαιτίας της αλλαγής της οργάνωσης.

 **Πλήθος εντολών:** Η μέτρηση του **πλήθους εντολών** για ένα πρόγραμμα μπορεί να γίνει αν έχουμε έναν μεταγλωττιστή για τον υπολογιστή μαζί με τα εργαλεία που υπολογίζουν τη συμπεριφορά του συνόλου εντολών .
Αν διαθέτουμε την μεταγλωττισμένη έκδοση του προγράμματος που μας ενδιαφέρει να μετρήσουμε, υπάρχουν δύο κύριες μέθοδοι που μπορούμε να εφαρμόσουμε για να υπολογίσουμε το πλήθος των εντολών. Στις περισσότερες περιπτώσεις θέλουμε να γνωρίζουμε εκτός από το σύνολο εντολών και την συχνότητα των διαφορετικών κλάσεων εντολών .
1ος τρόπος : Ο πρώτος τρόπος για να το υπολογίσουμε είναι να χρησιμοποιήσουμε έναν εξομοιωτή συνόλου εντολών ο οποίος μεταφράζει τις εντολές. Το κύριο μειονέκτημα αυτής της προσέγγισης είναι η ταχύτητα (εφόσον η προσομοίωση του συνόλου εντολών είναι αργή διαδικασία) και η ανάγκη για εφαρμογή ουσιωδών δομών, αφού για να χειριστεί ο εξομοιωτής μεγάλα προγράμματα θα πρέπει να υποστηρίζει συναρτήσεις του λειτουργικού συστήματος. 'Ενα πλεονέκτημα του εξομοιωτή συνόλου εντολών είναι ότι μπορεί να υπολογίσει με μεγάλη ακρίβεια σχεδόν κάθε περίπτωση συμπεριφοράς συνόλου εντολών όπως και λειτουργικού συστήματος. Μερικοί τυπικοί εξομοιωτές συνόλου εντολών τρέχουν 10 με 1000 φορές πιο αργά απ΄ ότι τρέχει το πρόγραμμα .

2ος τρόπος : Η εναλλακτική προσέγγιση χρησιμοποιεί παρακολούθηση κατά την διάρκεια της εκτέλεσης. Σύμφωνα με αυτή την προσέγγιση το δυαδικό πρόγραμμα τροποποιείται έτσι ώστε να συμπεριλάβει κώδικα ελέγχου (instrumentation code), όπως ένας μετρητής σε κάθε βασικό μπλοκ. Ενώ το πρόγραμμα τρέχει, καταγράφονται οι τιμές του μετρητή. Έπειτα προσδιορίζεται απλά η κατανομή των εντολών εξετάζοντας την στατική έκδοση του κώδικα και των τιμών των μετρητών. Οι μετρητές μας πληροφορούν για τη συχνότητα εκτέλεσης μιας εντολής. Οι τυπικοί κώδικες ελέγχου αυξάνουν τον χρόνο εκτέλεσης κατά 1,1 ή 2 φορές. Αυτή η τεχνική είναι πολύ γρήγορη, εφόσον το πρόγραμμα εκτελείται και δεν μεταφράζεται, και πολύ χρήσιμη όταν διαφέρει η αρχιτεκτονική της μηχανής που εξομοιώνεται με εκείνη της μηχανής που χρησιμοποιείται για την εξομοίωση. Σε αυτή την περίπτωση, το πρόγραμμα που ελέγχει τον κώδικα κάνει μία απλή μετάφραση μεταξύ των συνόλων εντολών. Αυτή η μετάφραση δεν είναι απαραίτητα και η βέλτιστη. Ακόμα και μία πρόχειρη μετάφραση συνήθως οδηγεί σε πιο γρήγορη μέτρηση του συστήματος από απ' ότι μία πλήρης εξομοίωση του συνόλου εντολών.

CPI: Η μέτρηση των κύκλων ρολογιού ανά εντολή (CPI) είναι δυσκολότερη, αφού εξαρτάται άμεσα απ' την αναλυτική περιγραφή της οργάνωσης του επεξεργαστή, καθώς επίσης και του **ρεύματος εντολών.** Για έναν πολύ απλό επεξεργαστή είναι δυνατό να υπολογίσουμε το CPI για κάθε εντολή από έναν πίνακα και απλά να πολλαπλασιάσουμε αυτές τις τιμές με τον αριθμό των εμφανίσεων του κάθε τύπου εντολών. Ωστόσο, αυτή η απλή προσέγγιση δεν λειτουργεί για τους περισσότερους σύγχρονους επεξεργαστές, οι οποίοι χρησιμοποιούν τεχνικές όπως η σωλήνωση και η ιεραρχία μνήμης. Κατά συνέπεια, οι εντολές δεν έχουν απλές μετρήσεις κύκλων αλλά εξαρτώνται από την κατάσταση του επεξεργαστή κατά την εκτέλεση της εντολής. Οι σχεδιαστές χρησιμοποιούν συχνά τις μέσες τιμές κύκλων ρολογιού ανά εντολή, αλλά αυτά τα μέσα CPI υπολογίζονται μετρώντας τα αποτελέσματα της σωλήνωσης και της δομής της κρυφής μνήμης.

Για να προσδιορίσουμε τον αριθμό κύκλων ρολογιού για μία εντολή σε έναν σύγχρονο επεξεργαστή, υποθέτουμε ότι έχουμε ένα τέλειο σύστημα μνήμης. Είναι συχνά χρήσιμο να διαχωρίζουμε εκείνα τα συστατικά μέρη που προκύπτουν από το σύστημα μνήμης και εκείνα που προσδιορίζονται από τον επεξεργαστή. Η χρησιμότητα οφείλεται:

1. στις διαφορετικές τεχνικές εξομοίωσης για την αξιολόγηση αυτών των συνεισφορών.
2. στο γεγονός ότι η συνεισφορά του συστήματος μνήμης έχει προστεθεί σαν μέσος όρος σε όλες τις εντολές, ενώ η συνεισφορά του επεξεργαστή είναι πιο πιθανό να καθοριστεί από τις εντολές.

Οι κύκλοι ρολογιού για κάθε εντολή i υπολογίζονται με την βοήθεια του τύπου 1.25:

$$\mathbf{CPI_I = CPI_I\ σωλήνωσης + CPI_I\ συστήματος\ μνήμης} \qquad \textbf{(τύπος 1.25)}$$

Οι κύκλοι ρολογιού για την σωλήνωση μοντελοποιούνται από την εξομοίωση της δομής της σωλήνωσης χρησιμοποιώντας το ρεύμα εντολών.

 Για απλές σωληνώσεις, είναι αποδοτικό να μοντελοποιήσουμε την απόδοση κάθε βασικού μπλοκ μεμονωμένα, αγνοώντας τις αλληλεπιδράσεις βασικών μπλοκ. Σε αυτές τις περιπτώσεις η απόδοση του κάθε βασικού μπλοκ, μαζί με την μέτρηση συχνότητας για κάθε βασικό μπλοκ, μπορεί να χρησιμοποιηθεί για να προσδιοριστεί το **ολικό CPI.** Το ολικό CPI περιλαμβάνει το μέσο αριθμό κύκλων ρολογιού ανά εντολή και το **CPI για κάθε εντολή**, δηλ. τους κύκλους ρολογιού για κάθε εντολή μεμονωμένα.

Χρήση της εξίσωσης απόδοσης της ΚΜΕ

Το πραγματικό μέτρο της απόδοσης ενός υπολογιστή είναι ο χρόνος. Έστω ότι αλλάζουμε το σύνολο εντολών για να μειώσουμε το πλήθος των εντολών. Η ενέργεια αυτή μπορεί να οδηγήσει σε μία οργάνωση με πιο αργό κύκλο ρολογιού ο οποίος αντισταθμίζει τις βελτιώσεις στο πλήθος των εντολών. Όταν συγκρίνουμε δύο υπολογιστές θα πρέπει να εξετάζουμε και τις τρεις συνιστώσες της εξίσωσης απόδοσης της ΚΜΕ για να καταλαβαίνουμε την σχετική απόδοση του υπολογιστή.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 4

- **Να απαριθμήσετε τις συνιστώσες της απόδοσης της ΚΜΕ.**
- **Να εξηγήσετε σε 10 το πολύ γραμμές τον τρόπο με τον οποίο γίνεται η μέτρηση των συνιστωσών αυτών; Στην περίπτωση που δυσκολεύεστε να απαντήσετε στις παραπάνω ερωτήσεις, κρίνεται σκόπιμο να επαναλάβετε την ανάγνωση της προηγούμενης υποενότητας με τίτλο: «Μέτρηση των συνιστωσών της απόδοσης της ΚΜΕ».**

Οι δείκτες της απόδοσης της ΚΜΕ : Επιτυχίες & Αποτυχίες

Όλα τα προηγούμενα χρόνια ήταν συνηθισμένο φαινόμενο η κάθε γενιά υπολογιστών να αχρηστεύει τις τεχνικές αξιολόγησης της απόδοσης της προηγούμενης.

Το αρχικό μέτρο σύγκρισης της απόδοσης ήταν ο χρόνος για να εκτελέσει μια ξεχωριστή λειτουργία, όπως η πρόσθεση. Εφόσον οι περισσότερες εντολές είχαν τον ίδιο χρόνο εκτέλεσης, η χρονομέτρηση μιας εντολής, μας έδινε πληροφορίες και για τις άλλες. Καθώς όμως οι χρόνοι εκτέλεσης των εντολών διαφοροποιήθηκαν, ο χρόνος μιας λειτουργίας δεν ήταν πια χρήσιμος για συγκρίσεις. Για να ληφθούν υπόψη αυτές οι διαφορές, ένας συνδυασμός εντολών υπολογίστηκε μετρώντας τη σχετική συχνότητα των εντολών σε έναν υπολογιστή μέσω πολλών προγραμμάτων. Πολλαπλασιάζοντας τον χρόνο για κάθε εντολή με το βάρος της εντολής στο μίγμα εντολών, έδωσε στον χρήστη τον μέσο χρόνο εκτέλεσης εντολής. Μιας και τα σύνολα εντολών ήταν παρόμοια, αυτή η σύγκριση ήταν πολύ πιο ακριβής από την πρόθεση χρόνων.

Αν μετριέται σε κύκλους ρολογιού, ο μέσος χρόνος εκτέλεσης εντολής είναι ίδιος με το μέσο CPI.

Καθώς οι κεντρικές μονάδες επεξεργασίας έγιναν πιο πολύπλοκες και βασίστηκαν περισσότερο σε ιεραρχίες μνήμης και στις σωληνώσεις, δεν υπήρχε πλέον ένας μόνο χρόνος εκτέλεσης ανά εντολή. Κατά συνέπεια, τα MIPS δεν μπορούσαν να υπολογιστούν από το μίγμα και το εγχειρίδιο. Το MIPS -Million Instructions Per Second, δηλώνει τα εκατομμύρια των εντολών ανά δευτερόλεπτο. Το επόμενο βήμα ήταν δοκιμές αποδόσεως χρησιμοποιώντας πυρήνες και συνθετικά προγράμματα.

Σύντομα διαπιστώθηκε ότι η χρησιμοποίηση των MIPS για την σύγκριση αρχιτεκτονικών με διαφορετικό σύνολο εντολών δεν θα πετύχει. Η λύση δόθηκε με τη δημιουργία του συγκριτικού MIPS. Ο VAX-11/780 θεωρήθηκε 1-MIPS υπολογιστής γιατί, συγκρινόμενος με τον IBM 370/158, "έτρεχε" τα προγράμματα με την ίδια ταχύτητα. Το συγκριτικό MIPS για μία μηχανή Μ ορίστηκε, βασισμένο σε μια μηχανή αναφοράς Μ, ως εξής:

(Τύπος 1.26)

$$\text{MIPS}_{\text{M}} = \frac{\text{Απόδοση}_{\text{M}}}{\text{Απόδοση}_{\text{αναφοράς}}} \times \text{MIPS}_{\text{αναφοράς}}$$

 Η δημοτικότητα του VAX-11/780 τον έκανε μηχανή αναφοράς για το συγκριτικό MIPS. Το γεγονός αυτό δεν μας εκπλήσσει, αφού το συγκριτικό MIPS για έναν 1- MIPS υπολογιστή υπολογίζεται εύκολα.

 Στα τέλη της δεκαετίας του 1980 ιδρύθηκε η SPEC (System Performance and Evaluation Cooperative) με σκοπό να βελτιώσει την κατάσταση στις δοκιμές της απόδοσης και να δημιουργήσει μια πιο έγκυρη βάση για συγκρίσεις. Η ομάδα αυτή αρχικά ασχολήθηκε αποκλειστικά με σταθμούς εργασίας και εξυπηρετητές στην αγορά των συστημάτων UNIX, πράγμα που ακόμα και σήμερα παραμένει το πρωταρχικό ενδιαφέρον αυτών των προγραμμάτων δοκιμής απόδοσης. Η πρώτη παρουσίαση των προγραμμάτων δοκιμής της SPEC, τα οποία τώρα λέγονται SPEC89, ήταν μία ουσιαστική βελτίωση στην χρήση πιο ρεαλιστικών προγραμμάτων δοκιμής απόδοσης. Τα SPEC89 αντικαταστάθηκαν από τα SPEC92. Αυτή η έκδοση μεγέθυνε το σύνολο των προγραμμάτων, έκανε μεγαλύτερες τις εισόδους σε κάποια προγράμματα δοκιμής απόδοσης και καθόρισε νέους κανόνες εκτέλεσης. Το 1994 η SPEC παρουσίασε κανόνες για μεταγλωττιστές (compilers) και διακόπτες μεταγλώττισης για να χρησιμοποιούνται στον καθορισμό της κατώτατης απόδοσης του SPEC92, με σκοπό την μείωση του μεγάλου αριθμού των ειδικών σημαιών μεταγλωττιστή για τα προγράμματα δοκιμής:

1. Οι επιλογές βελτιστοποίησης είναι ασφαλείς: είναι αναμενόμενο να μπορούν γενικά να χρησιμοποιηθούν σε κάθε πρόγραμμα.
2. Ο ίδιος μεταγλωττιστής και οι ίδιες σημαίες χρησιμοποιούνται σε κάθε πρόγραμμα δοκιμής απόδοσης.
3. Δεν επιτρέπονται σημαίες δηλώσεων που θα δώσουν στον μεταγλωττιστή κάποια δεδομένα τα οποία μπορεί να παράγει.
4. Σημαίες, οι οποίες επιτρέπουν εμβόλιμες ρουτίνες βιβλιοθήκης που συνήθως θεωρούνται μέρος της γλώσσας είναι επιτρεπτές, ενώ άλλες τέτοιες εμβόλιμες υποδείξεις δεν επιτρέπονται από τον 5ο κανόνα.
5. Στις σημαίες δεν επιτρέπονται ονόματα προγραμμάτων ή υπορουτινών.
6. Δεν επιτρέπονται βελτιώσεις βασισμένες στην ανατροφοδότηση
7. Σημαίες οι οποίες αλλάζουν το προεπιλεγμένο μέγεθος ενός αντικειμένου δεδομένων (π.χ απλή ακρίβεια σε διπλή ακρίβεια) δεν επιτρέπονται.

 **Επιτρεπτές είναι σημαίες οι οποίες κατευθύνουν τον μεταγλωττιστή να μεταγλωττίσει για μια ιδιαίτερη υλοποίηση και εκείνες που επιτρέπουν στον μεταγλωττιστή να χαλαρώσει ορισμένες αριθμητικές απαιτήσεις.**

Η πρόθεση τους ήταν να παραχθούν βασικά αποτελέσματα, ίδια με εκείνα ενός συνηθισμένου χρήστη, χωρίς να καταβάλει ιδιαίτερες προσπάθειες.
Επιπρόσθετα η SPEC επιδόθηκε στην παραγωγή προγραμμάτων δοκιμής απόδοσης κατευθυνόμενα από το σύστημα. Εφαρμόστηκαν για να δοκιμάσουν την απόδοση

συστημάτων με λειτουργίες I/O, λειτουργικού συστήματος και την μέτρηση της ρυθμαπόδοσης.

Ανάλυση της απόδοσης ανεξάρτητα από την υλοποίηση.

Καθώς ο διαχωρισμός μεταξύ της αρχιτεκτονικής και της υλοποίησης διαπέρασε την υπολογιστική κοινότητα την δεκαετία του 1970, δημιουργήθηκε το ερώτημα εάν η απόδοση μιας αρχιτεκτονικής από μόνη της μπορεί να αποτιμηθεί, εν αντιθέσει με την υλοποίηση μιας αρχιτεκτονικής. Τρία ποσοτικά μέτρα επινοήθηκαν για να εξετάσουν εξονυχιστικά τις αρχιτεκτονικές:

- S - Ο αριθμός των bytes του κώδικα ενός προγράμματος
- M - Ο αριθμός των bytes που μεταφέρονται μεταξύ της μνήμης και της CPU κατά την διάρκεια της εκτέλεσης για τον κώδικα και τα δεδομένα (το S μετρά το μέγεθος του κώδικα στο χρόνο μεταγλώττισης ενώ το M είναι την επικοινωνία με την μνήμη κατά την διάρκεια εκτέλεσης του προγράμματος)
- R - Ο αριθμός των bytes που μεταφέρονται μεταξύ καταχωρητών σε ένα κανονικό μοντέλο ΚΜΕ.

Η προσπάθεια να αποτιμηθεί η αρχιτεκτονική ανεξάρτητα από την υλοποίηση ήταν γενναία, όχι όμως και επιτυχής.

Η Τοπικότητα της αναφοράς και η απόδοση της ΚΜΕ

Η Τοπικότητα της αναφοράς

Παράλληλα με την εφαρμογή του νόμου του Amdahl σε όλα τα συστήματα, προέκυψαν και άλλες θεμελιώδεις παρατηρήσεις για τις ιδιότητες των προγραμμάτων. Η πιο σημαντική από αυτές είναι η ήδη γνωστή ως «**Τοπικότητα της αναφοράς**», κατά την οποία τα προγράμματα επιδιώκουν να ξαναχρησιμοποιήσουν δεδομένα και εντολές που πρόσφατα έχουν χρησιμοποιήσει. Είναι εμπειρικά αποδεδειγμένο ότι ένα πρόγραμμα αφιερώνει το 90% από τον εκτελέσιμο χρόνο του σε λιγότερο από το 10% του κώδικα. Απόρροια της τοπικότητας αποτελεί η πρόβλεψη της συχνότητας με την οποία θα επανεκτελεστούν τόσο εντολές όσο και δεδομένα (ανάκληση), λαμβάνοντας υπόψη τις προσπελάσεις του προγράμματος στο πρόσφατο παρελθόν. Ωστόσο η τοπικότητα της αναφοράς εφαρμόζεται για προσπελάσεις δεδομένων και όχι για προσπελάσεις κώδικα. Όπως προαναφέρθηκε, δύο είναι οι διαφορετικοί τύποι τοπικότητας, η τοπικότητα χώρου και η τοπικότητα χρόνου.

Η απόδοση της ΚΜΕ

Εξαιτίας της αρχής της τοπικότητας αλλά και της αυξημένης ταχύτητας των μικρότερων μνημών, η ιεραρχία μνήμης βελτιώνει σημαντικά την απόδοση. Υπάρχουν διάφοροι τρόποι για να επιβεβαιώσουμε την παραπάνω άποψη. Για την σύγκριση συστημάτων με ή χωρίς κρυφή μνήμη, καλύτερος είναι μέσω του νόμου του Amdahl**.**

Στην πράξη όμως για την αξιολόγηση της ιεραρχίας μνήμης χρησιμοποιούμε την επέκταση της εξίσωσης του χρόνου εκτέλεσης της ΚΜΕ. Αυτή περιλαμβάνει επιπρόσθετα τον αριθμό των κύκλων κατά τη διάρκεια των οποίων η ΚΜΕ καθυστερεί περιμένοντας μια πρόσβαση στη μνήμη και λέγονται **κύκλοι καθυστέρησης της μνήμης (memory stall cycles)**. Η απόδοση τότε δίνεται :

χρόνος εκτέλεσης της ΚΜΕ =(κύκλοι ρολογιού της ΚΜΕ + κύκλοι καθυστέρησης της μνήμης) *χρόνος ενός κύκλου ρολογιού

(Τύπος 1.27)

Στην εξίσωση αυτή υποθέτουμε ότι οι κύκλοι ρολογιού της ΚΜΕ περιλαμβάνουν το χρόνο για τη διαχείριση μιας επιτυχίας κρυφής μνήμης και το χρόνο που η ΚΜΕ καθυστερεί κατά τη διάρκεια μιας αποτυχίας κρυφής μνήμης . Ο αριθμός των κύκλων καθυστέρησης μνήμης εξαρτάται τόσο από τον αριθμό των επιτυχιών όσο και από το κόστος ανά αποτυχία το οποίο καλείται και **ποινή αποτυχίας (miss penalty)** και δίνεται από τον τύπο (1.28):

κύκλοι καθυστέρησης μνήμης = ***πλήθος αποτυχιών * ποινή αποτυχίας***
= IC * αποτυχίες ανά εντολή * ποινή αποτυχίας
= IC *αναφορές στη μνήμη ανά εντολή * ρυθμό αποτυχίας * ποινή αποτυχίας **(τύπος 1.28)**

(IC : instruction count)

 **Το πλεονέκτημα του τύπου (1.28), είναι ότι τα μέρη του υπολογίζονται εύκολα. Παραδείγματος χάριν ο ρυθμός αποτυχίας είναι το πλήθος των προσβάσεων που αποτυγχάνουν προς τις συνολικές προσβάσεις.**

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 5

Υποθέστε ότι η κρυφή μνήμη είναι 10 φορές πιο γρήγορη από την κύρια και ότι μπορεί να χρησιμοποιηθεί στο 90% του χρόνου . Να υπολογίσετε πόση επιτάχυνση πετυχαίνουμε με τη χρήση κρυφής μνήμης.

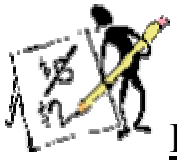
ΑΠΑΝΤΗΣΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ 5

Εφαρμόζοντας το νόμου του Amdahl, έχουμε:

$$\text{επιτάχυνση} = \frac{1}{(1 - \%\text{ του χρόνου χρήσης ΚΜΕ}) + \frac{\%\text{ χρόνος χρήσης κρυφής μνήμης}}{\text{επιτάχυνση χρησιμοποιώντας την κρυφή}}}$$

$$\text{Άρα: επιτάχυνση} = \frac{1}{(1 - 90\%) + 90\%/10} = \frac{1}{0{,}1 + 0{,}09} \cong 5{,}3$$

Ως εκ τούτου με τη χρήση της κρυφής μνήμης επιταχύνουμε το σύστημα μας κατά 5,3 φορές .



Παράδειγμα

Υποθέστε ότι υπάρχει ένας υπολογιστής που το CPI (κύκλοι ρολογιού ανά εντολή) είναι 2,0 όταν όλες οι προσβάσεις στη μνήμη επιτυγχάνουν στην κρυφή. Τα μόνα δεδομένα που προσπελαύνονται είναι φόρτωσης και αποθήκευσης και αποτελούν το 40% των εντολών. Αν η ποινή αποτυχίας είναι 25 κύκλοι ρολογιού και ο ρυθμός αποτυχίας 2% πόσο πιο γρήγορο θα ήταν το μηχάνημα αν όλες οι εντολές ήταν επιτυχίες κρυφής μνήμης ;

Απάντηση:

Πρώτα πρέπει να υπολογίσουμε την απόδοση της μηχανής που πάντα πετυχαίνει (hits).

χρόνος εκτέλεσης ΚΜΕ = (κύκλοι ρολογιού ΚΜΕ + κύκλοι καθυστέρησης μνήμης)*
χρόνος ενός κύκλου ρολογιού.
= (IC * CPI +0) * χρόνος ενός κύκλου ρολογιού
= IC * 2,0 * χρόνος ενός κύκλου ρολογιού

Για τη μηχανή με την πραγματική μνήμη , πρώτα πρέπει να υπολογίσουμε τους κύκλους καθυστέρησης μνήμης :

κύκλοι καθυστέρησης μνήμης = (IC * αναφορές στη μνήμη ανά εντολή *
ρυθμός αποτυχίας * ποινή αποτυχίας)
= IC * (1 + 0,4) *0,02 * 25
= IC *0,7

όπου ο όρος (1+0,4) υποδηλώνει μια πρόσβαση εντολής και 0,4 προσβάσεις δεδομένων ανά κύκλο ρολογιού . Άρα η ολική επίδοση είναι :

(χρόνος εκτέλεσης ΚΜΕ $_{\text{με κρυφή μνήμη}}$) = (IC *2,0 +IC * 0,7) * κύκλος ρολογιού
= 2,7 *IC * χρόνος ενός κύκλου ρολογιού

και αν πάρουμε το λόγο αυτών :

$$\frac{\text{χρόνος εκτέλεσης ΚΜΕ}_{\text{με κρυφή μνήμη}}}{\text{χρόνος εκτέλεσης ΚΜΕ}} = \frac{\text{2,7 * IC * χρόνος ενός κύκλου ρολογιού}}{\text{2,0 * IC * χρόνος ενός κύκλου ρολογιού}}$$

$$= 1{,}35$$

Άρα η μηχανή που δεν έχει αποτυχίες κρυφής μνήμης είναι 1,35 φορές γρηγορότερη.

Ανακεφαλαιώνοντας λοιπόν …

Η θεμελιώδης αρχή σχεδιασμού, η βελτίωση της συνολικής απόδοσης, εξαρτάται κυρίως από τη βελτιστοποίηση της μέσης και πιο συχνής περίπτωσης.

Ο νόμος του Amdahl δίνει τον ορισμό της επιτάχυνσης και χρησιμοποιείται για τον υπολογισμό του κέρδους στην απόδοση, όταν βελτιστοποιηθεί ένα τμήμα του υπολογιστή.

Απ' την εξίσωση απόδοσης της ΚΜΕ, φαίνεται ότι αυτή εξαρτάται από 3 χαρακτηριστικά:

- **Τον κύκλο ή συχνότητα ρολογιού του υπολογιστή**
- **Τον αριθμό κύκλων ανά εντολή**
- **Το πλήθος εντολών**

Για να προσδιορίσουμε την απόδοση αξιοποιώντας την εξίσωση απόδοσης της ΚΜΕ, χρειαζόμαστε μετρήσεις για κάθε μία από τις συνιστώσες της εξίσωσης απόδοσης της ΚΜΕ: τον κύκλο ρολογιού, το πλήθος εντολών και τους κύκλους ρολογιού ανά εντολή.

Κάθε γενιά υπολογιστών χρησιμοποιεί διαφορετικά μέτρα σύγκρισης για να αξιολογήσει την απόδοσή της. Ξεκινήσαμε με τον χρόνο εκτέλεσης μιας ξεχωριστής λειτουργίας, το μέσο χρόνο εκτέλεσης, τους πυρήνες, τα συνθετικά προγράμματα καταλήγοντας στην έννοια του συγκριτικού MIPS. Κατά την αποτίμηση αρχιτεκτονικών ανεξάρτητα της υλοποίησης, επινοήθηκαν τρία ποσοτικά μέτρα – S, M, R σημαίες - χωρίς ωστόσο ιδιαίτερη επιτυχία.

Η τοπικότητα της αναφοράς παρέχει πολύτιμες πληροφορίες για την αξιοποίηση εντολών και δεδομένων. Η αξιολόγηση της ιεραρχίας της μνήμης γίνεται με την επέκταση της εξίσωσης για τον υπολογισμό του χρόνου εκτέλεσης της ΚΜΕ, έτσι ώστε να περιλαμβάνει τον αριθμό των κύκλων κατά τη διάρκεια των οποίων η ΚΜΕ καθυστερεί περιμένοντας μια πρόσβαση στη μνήμη.

Συγκρίνουμε ένα σύστημα με ή χωρίς κρυφή μνήμη με εφαρμογή του νόμου του Amdahl. Εναλλακτικά χρησιμοποιείται ο ρυθμός αποτυχίας, δηλαδή το πλήθος των αποτυχημένων προσβάσεων προς τις συνολικές προσβάσεις.