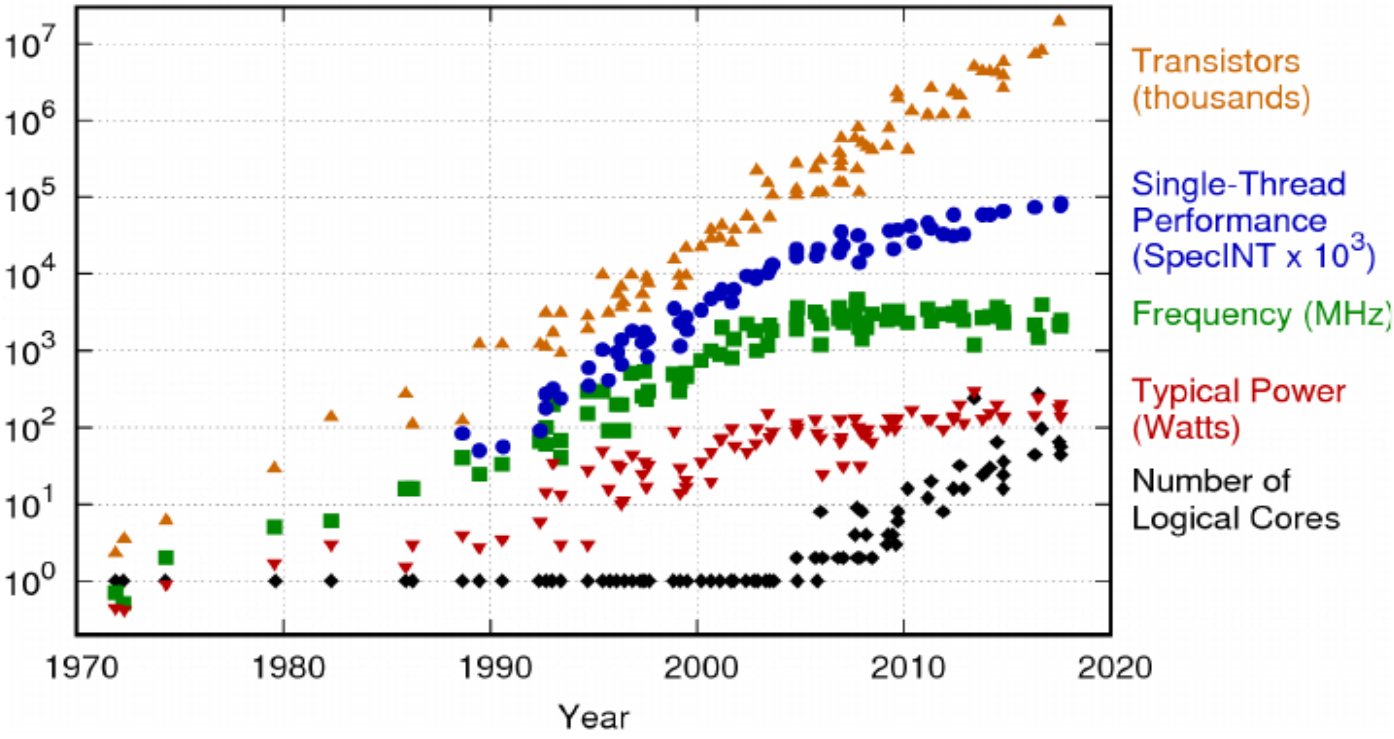
Αρχιτεκτονικη Υπολογιστων

1)



Για αρχή παρατηρούμε πώς με τα χρόνια έγινε μεγαλυ αύξηση στην χρήση των transistor, αλλά μια σχεδόν σταθερή στάση στο νούμερο των logical cores. Λογού του νομού του Moore η επίδραση που είχαν τα transistor στην αρχιτεκτονική υπολογιστών ήταν αρκετά μεγάλη καθώς έπρεπε να αλλάξουν την λογική τους. Έγινα πιο power efficient και επίσης αρκετά πιο γρήγορα τα μηχανήματα και γιαυτό η πολυπλοκότητα και ο αριθμός των register αυξήθηκαν. Η αρχιτεκτονική υπολογιστών επηρεάζεται σημαντικά από της τάσεις και δυνατότητες των τεχνολογιών

2)

αρχιτεκτονική υπολογιστών:Η αρχιτεκτονική υπολογιστών είναι το σύνολο των πράξεων, και κανόνων που χρησημοποιουνται για την λειτουργικότητα των υπολογιστικών συστημάτων
μικρο-αρχιτεκτονικής:Η μικρο αρχιτεκτονικη ειναι η οργανωση υπολογιστων,οπως και επισης ειναι ο τροπος με τον οποιο ενα συστηματα οδηγιων εφαρμοζονται στον ιδιο τον επεξεργαστη
Instruction Set Architecture: Ένα σύστημα ISA, είναι οι υποστηριζόμενοι τύποι δεδομένων, τα registers και η υποστήριξη υλικού για την διαχειρηση της κυρίας μνήμης. Σε γενικές γραμμές είναι ένα αφηρημένο μοντέλο ενός υπολογιστή

i) την ίδια αρχιτεκτονική :

DESIGNER	FAMILY	MODEL	MICRO
CENTAUR	WINCHIP	180	WINCHIP
CENTAUR	WINCHIP	200	WINCHIP
CENTAUR	WINCHIP	240	WINCHIP

ii) την ίδια αρχιτεκτονική αλλά διαφορετική μικρο-αρχιτεκτονική:

DESIGNER	FAMILY	MODEL	MICRO
INTEL	ATOM	BAY TRAIL	Z3740
INTEL	ATOM	BAY TRAIL	Z3770
INTEL	ATOM	BAY TRAIL	Z3795

Βλέπουμε πως παρόλο οι επεξεργαστές έχουν ίδια αρχικετονικη, αλλάζει η μικροαρχιτεκτονικη τους, αυτή η μεγάλη διάφορα έχει γίνει κυρίως για τον λόγω που χρησημοποιουνται δηλαδή θέλουν ο κάθε επεξεργαστής να κάνει complie διαφορετικές εντολές

3)

I)

Ξέρουμε οτι IPC= INSTRUCTIONS/SECONDS
Επομένως για το S1 έχουμε: 2 * 10^6 :IPC
Επομένως για το S2 έχουμε: 3.2 * 10^6 :IPC

ii)

Ξέρουμε οτι το CPI = CLOCK CYCLES / INSTRUCTIONS
Αρα δεδομένου οτι έχουμε για
S1: 20 MHZ
S2: 30 MHZ
Επομένως για να βρουμε το CPI για το S1 έχουμε:
(20 MHZ/ 2 * 10^6) = 10

Για να βρουμε το CPI για το S2 έχουμε:
(30 MHZ/ 3.2 * 10^6) = 9.3

iii)

Για να βρούμε το IPC σε κάθε μήχανη έχουμε
CLOCK * CPI

Επομένως για να βρουμε το IPC του S1 έχουμε:
(2 * 10^6)* 10 = 20 * 10^6
Ξέρουμε οτι IPC= INSTRUCTIONS/SECONDS

Αρα για S1 έχουμε: 6.6 * 10^6 :IPC

Επομένως για να βρουμε το IPC του S2 έχουμε:
(3 * 10^6)* 9.3 = 27.9* 10^6
Ξέρουμε οτι IPC= INSTRUCTIONS/SECONDS
Αρα για S2 έχουμε: 6.9 * 10^6 :IPC

4)

```
.data
A: .word 0,0,0,0,0,0

.text

LA $t0, a
ADDI $t1, $t1, 1
ADD $t4, $t1, $zero
SW $t1, 0($t0)

ADDI $t3, $t3, 6
ADDI $t2, $t2, 1

LOOP:
BLE $t2, $t3, LOOP_BODY
J EXIT_LOOP

LOOP_BODY:
ADDI $t1, $t1, 4
ADD $t4, $t4, $t1
ADDI $t0, $t0, 4
SW $t1, 0($t0)
ADDI $t2, $t2, 1
J LOOP

EXIT_LOOP:
```

I)

LA,ADDI,ADD,SW,LOOP,BLE
Αυτές είναι ολες οι εντολές που χρησημοποιούντε απο το πρόγραμμα , σε σύνολο είναι δηλαδή 6

ii)

Το instruction trace του προγραμματος είναι 16 εντολές

iii)

CPI = CLOCK CYCLES / IPC
IPC = INSTRUCTIONS / SECONS
IPC = 16 / 0.0005 = 32,000
Αρα
CPI = 20 MHZ / 32000 = 625

5)

Instruction class	CPI of the Instruction class
A	1
B	3
C	4

Code sequence	Instruction count (in millions)		
	A	B	C
1	2	1	2
2	4	3	1

I)

	A	B	C
CPI	1	3	4
CLOCK	2 MHZ	3 MHZ	8 MHZ

Αρα το CPI για το Sequence 1 :
CPI = CLOCK / INSTRUCTIONS
A: 2 * 10^6 / 2 * 10^6 = 1 CPI
B: 3 * 10^6 / 1 * 10^6 = 3 CPI
C: 8 * 10^6 / 2 * 10^6 = 4 CPI

	A	B	C
CPI	1	3	4
CLOCK	4 MHZ	9 MHZ	4 MHZ

Αρα το CPI για το Sequence 2 :
CPI = CLOCK / INSTRUCTIONS
A: 4 * 10^6 / 4 * 10^6 = 1 CPI
B: 9 * 10^6 / 3 * 10^6 = 3 CPI
C: 4 * 10^6 / 1 * 10^6 = 4 CPI

ii)

Για να βρούμε το IPC σε κάθε μήχανη έχουμε
CLOCK * CPI

IPC:	A	B	C
S1	2 * 10^6	9 * 10^6	24 * 10^6
S2	4 * 10^6	27 * 10^6	16 * 10^6

Παρατηρούμε ότι η 2η ακολουθεια κώδικα είναι πιο γρήγορη σε 2 instruction classes εννοώ η 1η ακολουθεια μόνο σε 1. Επομένως μπορούμε να πούμε ότι η 2η ακολουθέια κωδικά είναι ταχύτερη

6)

Έχουμε
clock = 500 MHZ
2 second execution time

Ξερουμε οτι IPS = INSTRUCTIONS / SECONDS
IPS * SECONDS = INSTRUCTIONS * 1
500 * 2 * 10^6 = 1000 * 10^6

I)

Επομένως για 1,2 ghz εχουμε:
seconds = instructions / IPS =

1200 * 10^6 / 1000 *10^6 = 1.2s
Αρα το σύστημα θα πάρει 1.2 seconds

ii)

Εφόσον το σύστημα αφιερωνει 40% της εκτέλεσεις σε εντολες τυπου add , αλλα με την καινουργια αρχιτεκτονικη οι εντολες τυπου add γίνονται 12 φορες πιο γρήγορα θα έχουμε:

για 500 MHZ: 1.3s

για 1.200 MHZ: 0,8s

7)

```
.data                           # 217129 Μαυροπουλος Ανδρεας
       A: .word 5,3             # 2D Array
          .word 3,9
       size: .word 2            # Size of 2d array
       .eqv Upper_Bound  10     # Upperbound constant
       .eqv DATA_SIZE 4         # Data size (4 for int)
.text
   main:
   la $a0 , A
   lw $a1 , size
   addi $a2,$zero,Upper_Bound          # a2 = uuper bound
   li $t0,0                            # t0 = index
   jal sumRow
   move $a0,$v0                        # v0 has the sum
   li $v0,1                            # Print out the value
   syscall
   # End the program
   li $v0,10
   syscall

  sumRow:
      li $v0,0         # sum = 0
       fLoop:
               mul $t1,$t0,$a1                # t1 = row_index * col_size
               add $t1,$t1,$t0                # + coll_index
               mul $t1,$t1,DATA_SIZE          # + col_size
               add $t1,$t1,$a0                # + base_address

               lw $t2,($t1)
               add $v0,$v0,$t2                # sum = sum + A[i][i]
               addi $t0,$t0 ,1
               blt $t0,$a1,fLoop              # Run the loop while index is smaller than the size
               slt $s0,$a2,$v0                # Check if sum is bigger than the upper bound
               beq $s0,1, ZeroLoc
       ZeroLoc:
       move $v0,$zero                 # Make location 0

   jr $ra

```