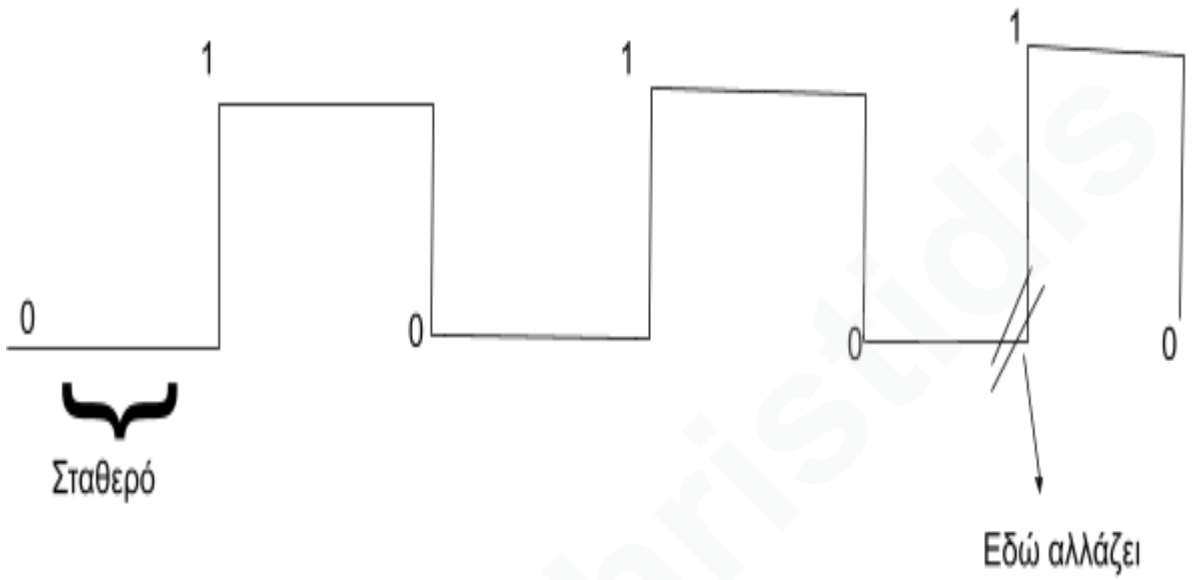
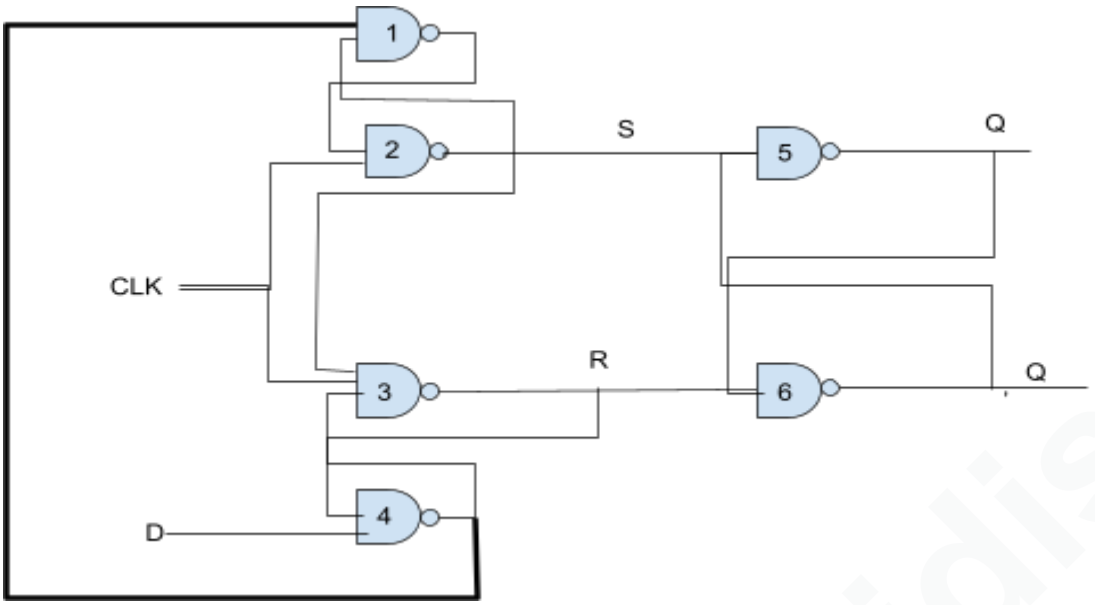
# Αρχιτεκτονική Διλεξη 5

*Flip-Flop και Clock*

CLK:Γεννήτρια παλμών σταθερής διάρκειας




1)Το κύκλωμα μπορεί να αλλάξει κατάσταση (το αποθηκευμένο Bit )μόνο κατά τη χρονική στιγμή που το ρολόι μεταβαίνει στο λογικό 1 Όσο το ρολόι Είναι μηδέν ή ένα σταθεροποιημένο το FF δεν αλλάζει κατάσταση.
Ανα τακτά χρονικά διαστήματα ΜΠΟΡΕΊ να αλλάξει το Flip Flop
2)Μία είσοδος D(data) είναι αυτή που τροφοδοτεί το κύκλωμα με το Bit που θα αποθηκεύσει.




Αν CLK=0, τότε οι έξοδοι των αυλών 2,3 είναι 1 δηλαδή S=R=1 Άρα τα Q,Q' διατηρούν την κατάστασή τους (ηρεμία)

η χοντρή μαύρη γραμμή χρειάζεται γιατί αν τυχόν αλλάξει το δόντι τότε δεν θα επηρεάσει μέσα τη μνήμη καθώς θα αλλάξει μόνο όταν το ρολόι μεταβεί από το 0 στο 1

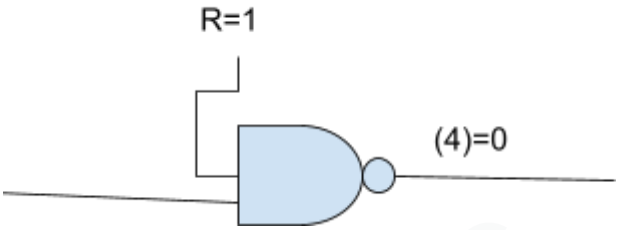
Αν δεν υπήρχε αυτό θα χάναμε δεδομένα

1) Αν CLK=0 (ηρεμία)
2) Οταν CLK ↗ τότε Q=D
3) Οσο CLK=1,Ακόμη και αν αλλάζει το D, η τιμή δεν αποθηκεύεται δηλαδή το FF δεν αλλάζει η κατάσταση παρά μόνο στον επόμενο θετικό παλμο

1) t=0, CLK=0,Q=0,Q'=1
   Επειδή CLK=0 την στιγμή t=0 θα είναι D=1 αλλα Q=0

2) t=5, CLK=1,D=1,S=R=1
Το ρολοι παράγει παλμούς κάθε 5 χρονικές μονάδες

Βήμα 1) Επειδή D=R=1 η (4)=0
Βήμα 2) Επειδή (4)=0, η (1)=1
Βήμα 3)Επειδή CLK=1 και (1)=1
το S=0
Βήμα 4) Επειδή S=0 το Q=1
Βήμα 5) Q=R=1, το Q'=0
→Q=D=1



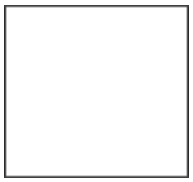

Κάθε χρονική στιγμή που το D γίνεται από 0→1 . Έχουμε Q = D

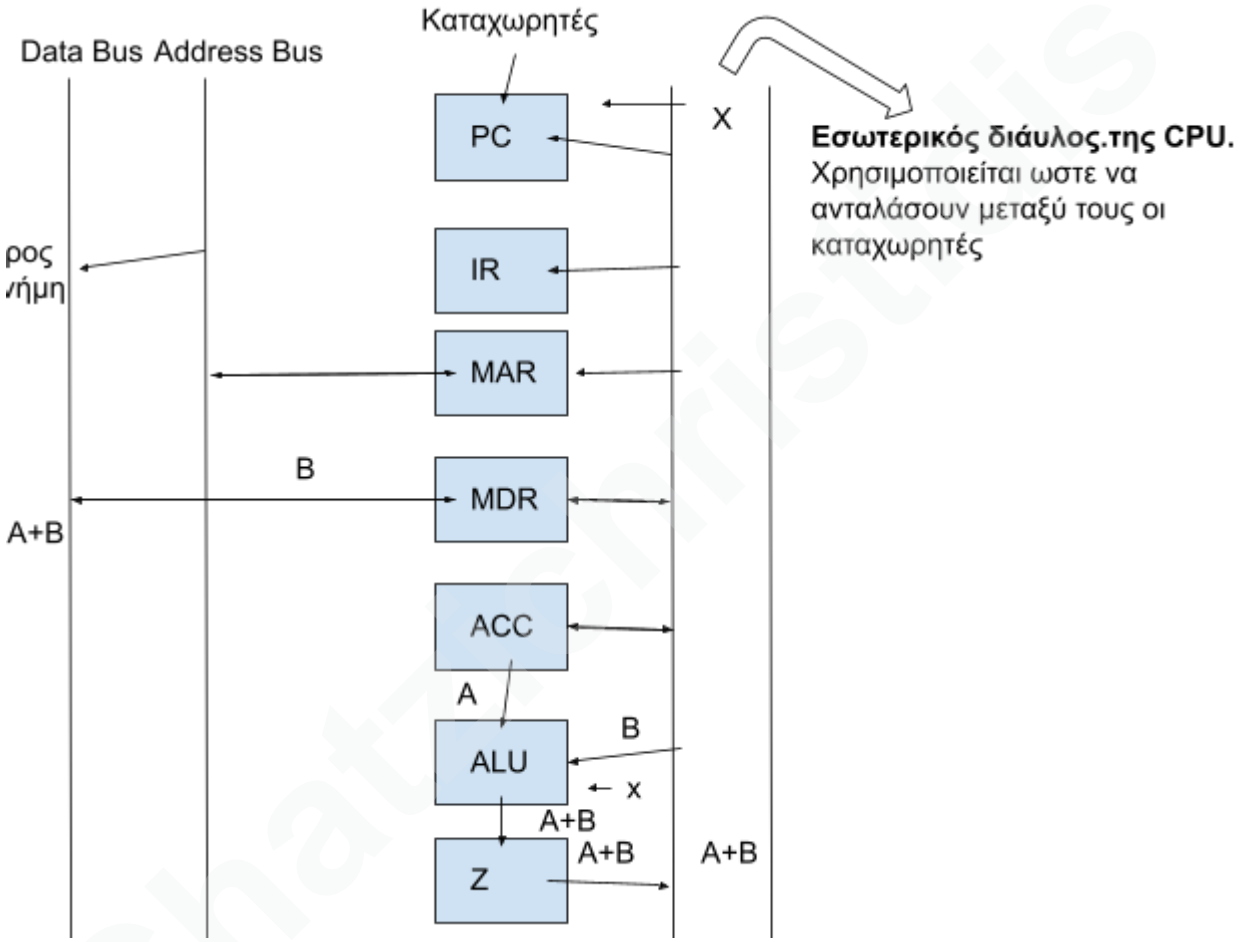
3) t=7 (πρίν τον επόμενο παλμό) το D γίνεται → D=0
Τι γίνεται στο Q;
Βήμα 1) D=0 → η (4) = 1 (απο NAND) R=1
Βήμα 2) (1) Επειδή S=0 θα δώσει έξοδο 1 (1)=1
Βήμα 3) Επειδή (1)=CLK=1 → S=0 αρα Q=1. Αρα η τιμή του D δεν περνάει στην έξοδο
Βήμα 4) t=10, CLK=0 → (S=R=1) →ηρεμία
Βήμα 5) t=15,D=0 όμως S=R=1
οπότε με παρόμοια ανάλυση όπως στο βήμα 2 (t=5) θα είναι Q=D=0

Συνοπτικά
1)Οταν CLK ↗ τοτε Q=D
2) Οσο CLK=1 καμία αλλαγή
3) Οταν CLK ↘ (ηρεμία)

Καταχωρητές CPU:Βοηθητική μνήμη της CPU

Register 8 bit:8 FF συγχρονισμένα με ένα κοινό ρολόι






PC(Programm Counter) μετρητής προγράμματος
Αποθηκέυει την διέυθυνση της επόμενης προς εκτελεση εντολής

Εκτελείται → Εντολή 1
PC → Εντολή 2

PC: αποθηκεύει διευθυνσεις εντολών

IR: Instruction Register / Καταχωρητής εντολών

Εστω μια CPU διαθέτει 8 εντολές. Πρέπει για να ξέρει η CPU τι θα κάνει, κάθε εντολή να έχει έναν κωδικό (OP CODE :Operation CODE)
ο IR κραταει κωδικο εντολής.
IR→ 3 BIT

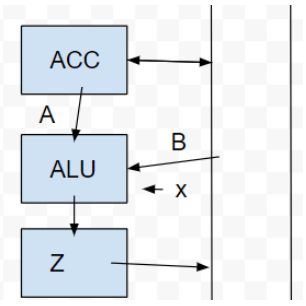
| Opcode(τα κραταει ο IR) | Εντολες CPU(παραδείγματα) |
|---|---|
| 000 | A+B |
| 001 | A:B |
| 010 | A*B |
| 011 | Φορτωση μιας λέξης απο CPU→RAM |

MAR: Στενή διευθύνσεις για αποκωδικοποίηση από τη μνήμη
MDR:Στέλνει δεδομένα στη μνήμη και παίρνει δεδομένα από αυτήν

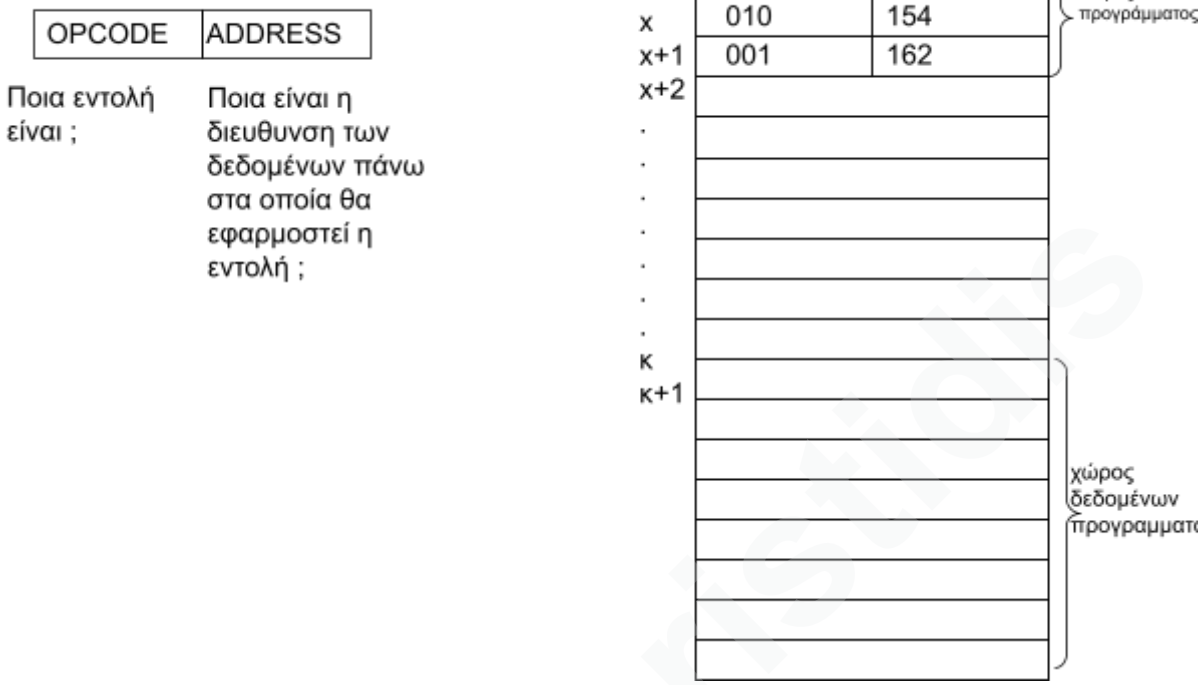
***Παράδειγμα:***
A+B:$T_A$ A και B ερχονται απο την μνήμη

**1)**Διαβάζεται το A από τη μνήμη και πάει στον MDR στην συνέχεια πρέπει να φέρει το B Άρα αποθηκεύει το A σε εναν καταχωρητη τον ACC(Accumulator)
**2)** Φέρνει το B

ALU(Arithmetic Logic Unit) περιέχει κυκλώματα που κάνουν πράξεις (Αθροιστής, Αφαιρέτης,...)
**3**) Το A πάει από τον ACC→ ALU και από εκεί στον αθροιστή. Το B θα πάει στην ALU με εσωτερικού διαύλου. Όπως δείχνει το σχήμα →
4) Η ALU δεν έχει μνήμη Άρα όταν ο αθροιστής βγάλει αποτέλεσμα το αποθηκεύει στο Z
5) Το Z βάζει το αποτέλεσμα A+B στον διάυλο το λαμβάνει ο MDR και το περνάει στην μνήμη



Εστω 8 εντολές αρα OPCODE, 3 bit διευθυνση μνήμη




***Ανάκληση :***

Κάθε εντολή περνάει από 2 στάδια:

1)Ανάκληση : Φέρνω μια εντολή στη CPU απο την μνήμη για να την εκτελέσω. Επειδή η CPU δεν ξέρει ποια εντολή θα έρθει προς εκτέλεση η ανάκληση είναι η ΙΔΙΑ για όλες τις εντολές

Έστω ότι εκτελείται η εντολή που βρίσκεται στη διεύθυνση x-1. Αυτό σημαίνει ότι ο PC δείχνει → X (στην επόμενη)

Για να ανακαλέσουμε την εντολή της διεύθυνσης X πρέπει να δώσουμε για αποκωδικοποίηση τι διεύθυνση X στον MAR .

MAR ←PC

Οταν συμβεί αυτό ο MDR θα πάρει τα δεδομένα από τη θέση μνήμη X για να τα φέρει στην CPU MDR←M[MAR]

Ο IR θα πρέπει να πάρει απο τον MDR το OPCODE

IR ← MDR[OPCODE]

Αυξηση του PC ώστε να δείχνει στην εντολή της διεύθυνσης X+1

Ανακληση (Βημα-Βημα) (κοιτάω σχήμα)

T0:MAR←PC
T0:Ο PC βάζει μια τιμή στον διαυλο. Η τιμή αυτή μπορεί να διαβαστεί απο όλους τους καταχωρητές
T1:Αποκωδικοποιειται η διευθηνση X
MDR← 010 154
T2: IR ← 010

| | |
|---|---|
| T0:MAR←PC,Z←PC+1 | Γράφει στο διαυλο της CPU αυτός ο καταχωρητής που είναι από δεξιά. Η προσθεση χρησιμοποιει την ALU |
| T1:MDR←M[MAR],PC←Z | Η πράξη MDR←M[MAR] γίνεται εξωτερικά ΑΡΑ μπορεί ο Z να γράφει στον PC αφού είναι ελεύθερος ο διαυλος |
| T2:IR←MDR[OPCODE] | Στον χρονο T2 ο διαυλος χρησιμοποιειται απο τον MDR . Η μονάδα ελέγχου δίνει εντολές σε κάθε Reg για το πότε πρέπει να γράψει στον δίαυλο η να διαβάσει από αυτόν. |

T0: Δίαυλος χρησιμοποιείται από τον καταχωρητή που βρίσκεται στο δεξί μέλος της σχέσης ← . Εδώ ο PC. Η τιμή του PC διαβάζεται από το MAR για να αποκωδικοποιηθεί η διεύθυνση X. Η τιμή μέσω διαύλου περνάει στην ALU παει στον αθροιστή, αυξάνεται κατά 1 και αποθηκεύεται το Z
Z←ALU ΛΑΘΟΣ❌
Z←PC+1 ΣΩΣΤΟ ✔

T1: Ο MDR παίρνει τη λέξη από τη μνήμη. Εδώ χρησιμοποιείται ο εξωτερικός δίαυλος δεδομένων. Άρα ο εσωτερικός δίαυλος μπορεί να χρησιμοποιηθεί από τον Z για να δώσει στον PC την τιμή του X+1. Αρα ο PC παίρνει την αυξημένη του τιμή στο $2^{o}$ βήμα T1

T2: Ο εξωτερικός δίαυλος χρησιμοποιειται απο τον MDR στο βήμα T2 η CPU ξέρει ποια εντολή θα εκτελεσει και πάνω σε ποιά δεδομένα.