ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Πρόγραμμα Μεταπτυχιακών Σπουδών «Πληροφορική και Εφαρμογές»

Αρχές Ψηφιακής Τεχνολογίας

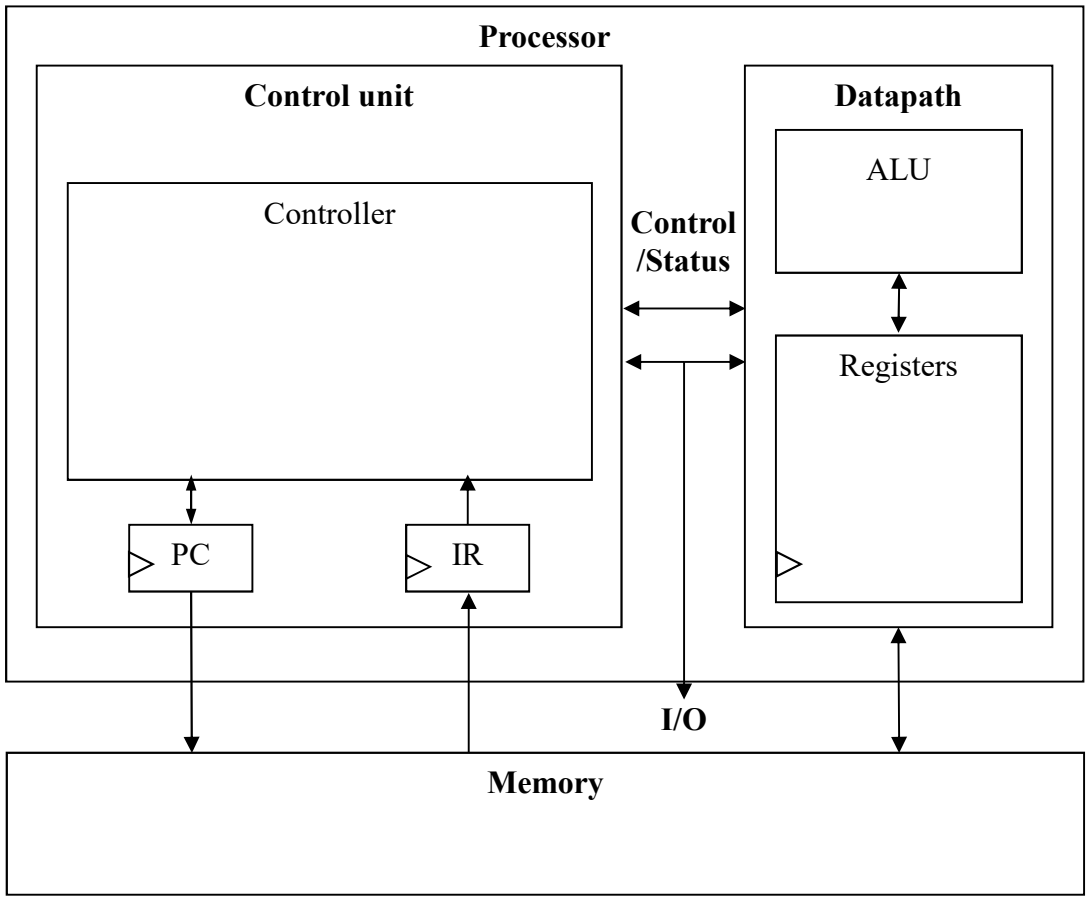
Επεξεργαστές γενικού σκοπού

Γιάννης Βογιατζής

2018-2019

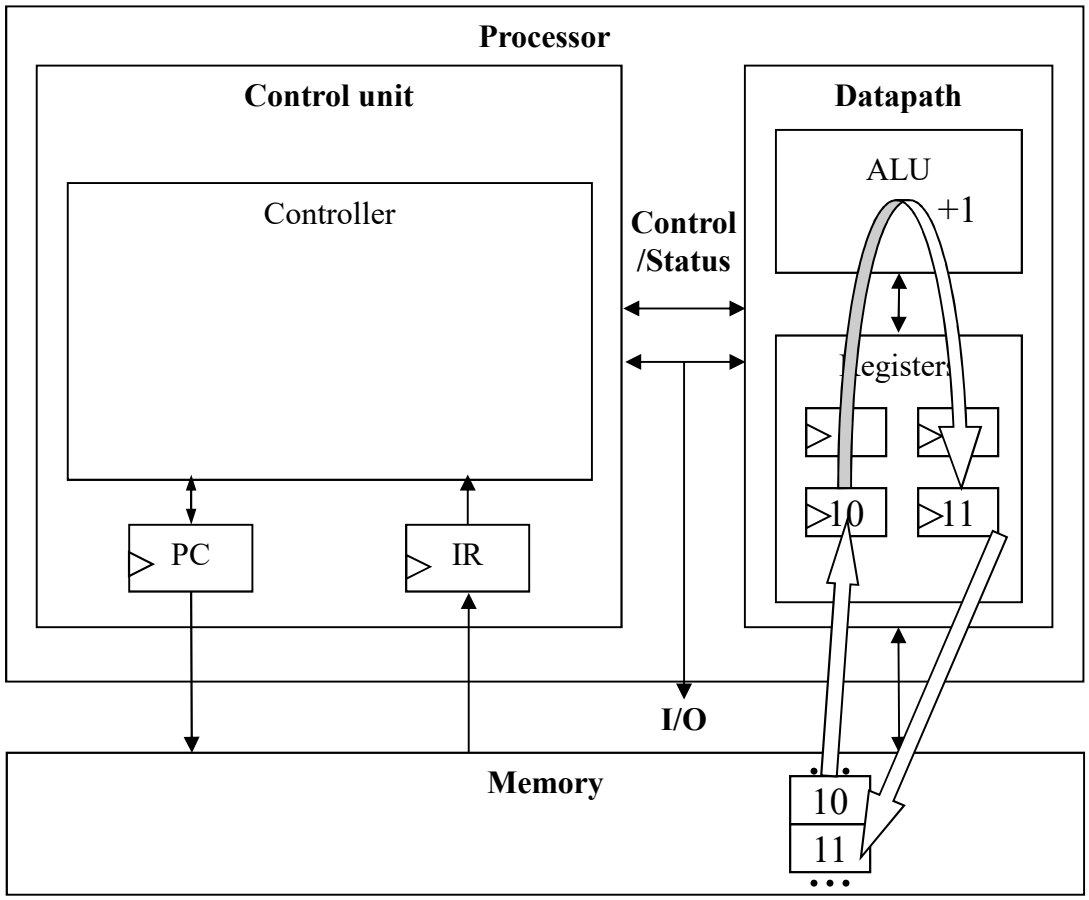
Βασική αρχιτεκτονική

- Control unit και datapath
  - Το datapath είναι γενικό
  - Η μονάδα ελέγχου δεν αποθηκεύει τον αλγόριθμο.
  - Ο αλγόριθμος προγραμματίζεται στη μνήμη



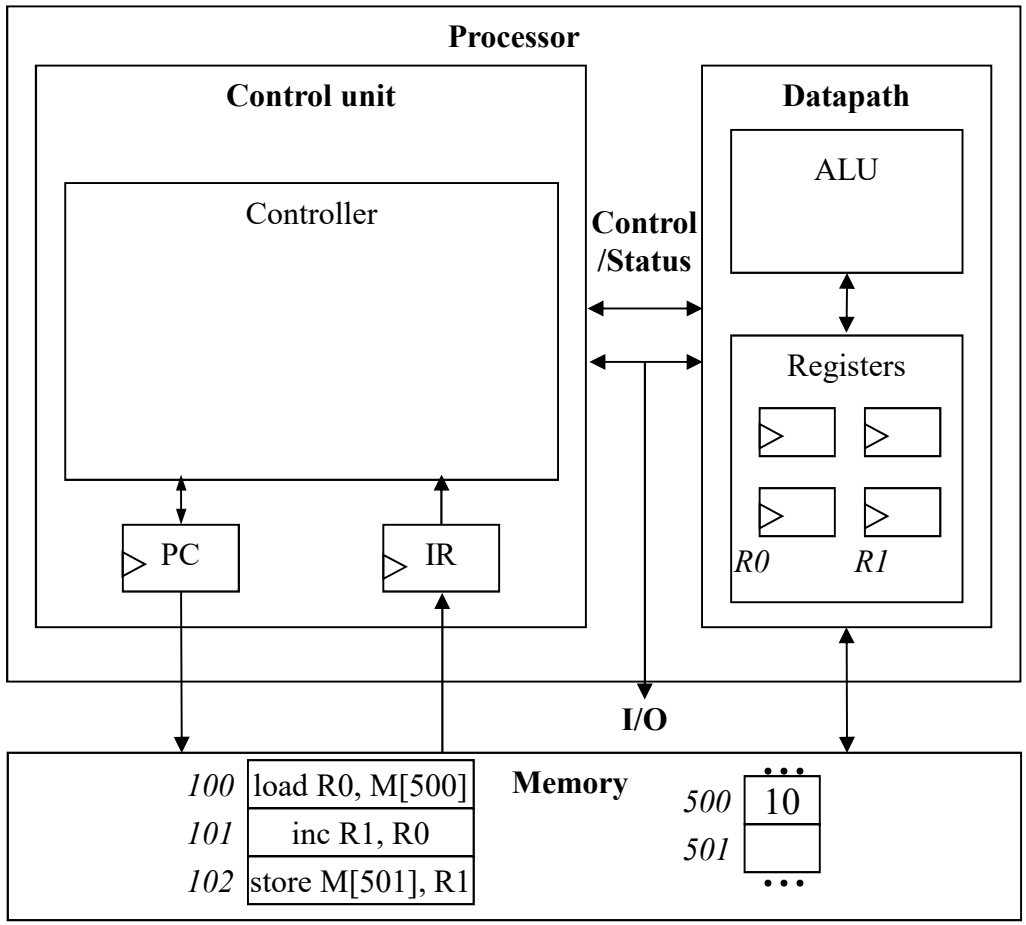
Λειτουργίες Datapath

- Load
  - Read memory location into register
- ALU operation
  - Input certain registers through ALU, store back in register
- Store
  - Write register to memory location



Control Unit

- Control unit: διαμορφώνει τις λειτουργίες του datapath
  - Οι εντολές αποθηκεύονται στη μνήμη
- Κύκλος εντολής – σπάει σε sub-operations, κάθε μία σε ένα κύκλο, π.χ:
  - Fetch: φέρε επόμενη εντολή από IR
  - Decode: αποφασίζω τι σημαίνει η εντολή
  - Fetch operands: μεταφορά δεδομένων από τη μνήμη
  - Execute: μετακίνηση δεδομένων μέσα στην ALU
  - Store results: αποθήκευση δεδομένων από τους καταχωρητές στη μνήμη



Απλό σύνολο εντολών

Assembly instruct.	First byte		Second byte		Operation
MOV Rn, direct	0000	Rn	direct		Rn = M(direct)
MOV direct, Rn	0001	Rn	direct		M(direct) = Rn
MOV @Rn, Rm	0010	Rn	Rm		M(Rn) = Rm
MOV Rn, #immed.	0011	Rn	immediate		Rn = immediate
ADD Rn, Rm	0100	Rn	Rm		Rn = Rn + Rm
SUB Rn, Rm	0101	Rn	Rm		Rn = Rn - Rm
JZ Rn, relative	0110	Rn	relative		PC = PC+ relative (only if Rn is 0)
	opcode		operands		

Q1: Πόσες εντολές μπορούμε να έχουμε;

Q2: Πόσους καταχωρητές μπορούμε να έχουμε;

Q3: Πόση μνήμη μπορούμε να δούμε;

Απλό πρόγραμμα

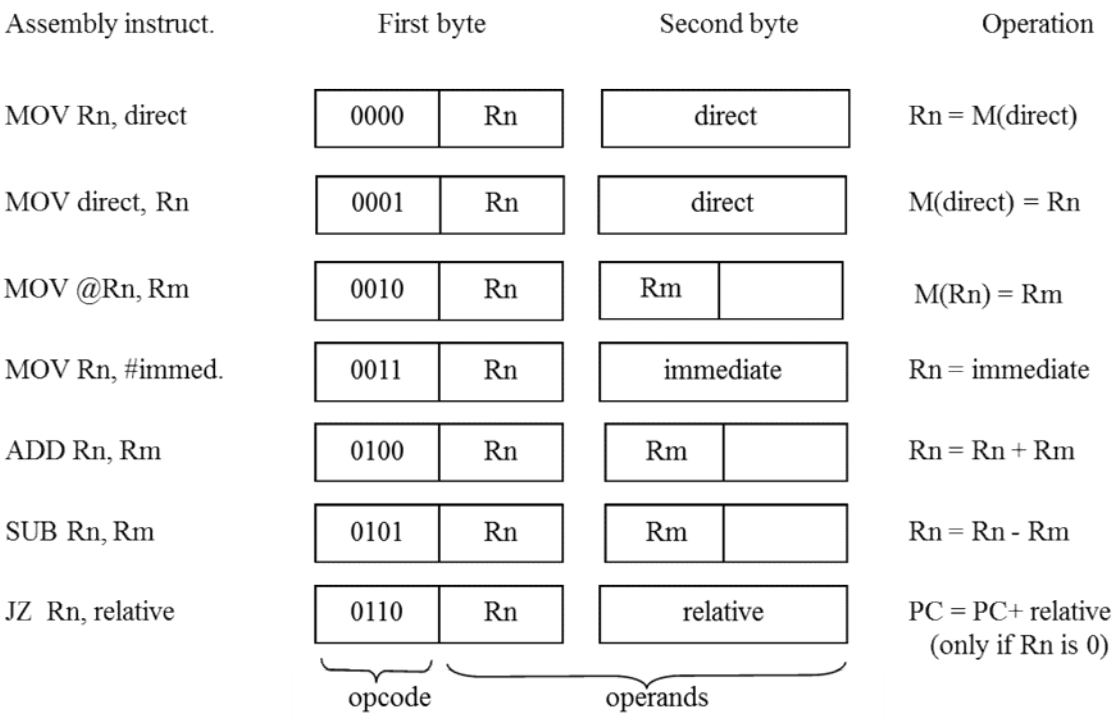
C program

```
int total = 0;
for (int i=10; i!=0; i--)
   total += i;
// next instructions...
```

Equivalent assembly program

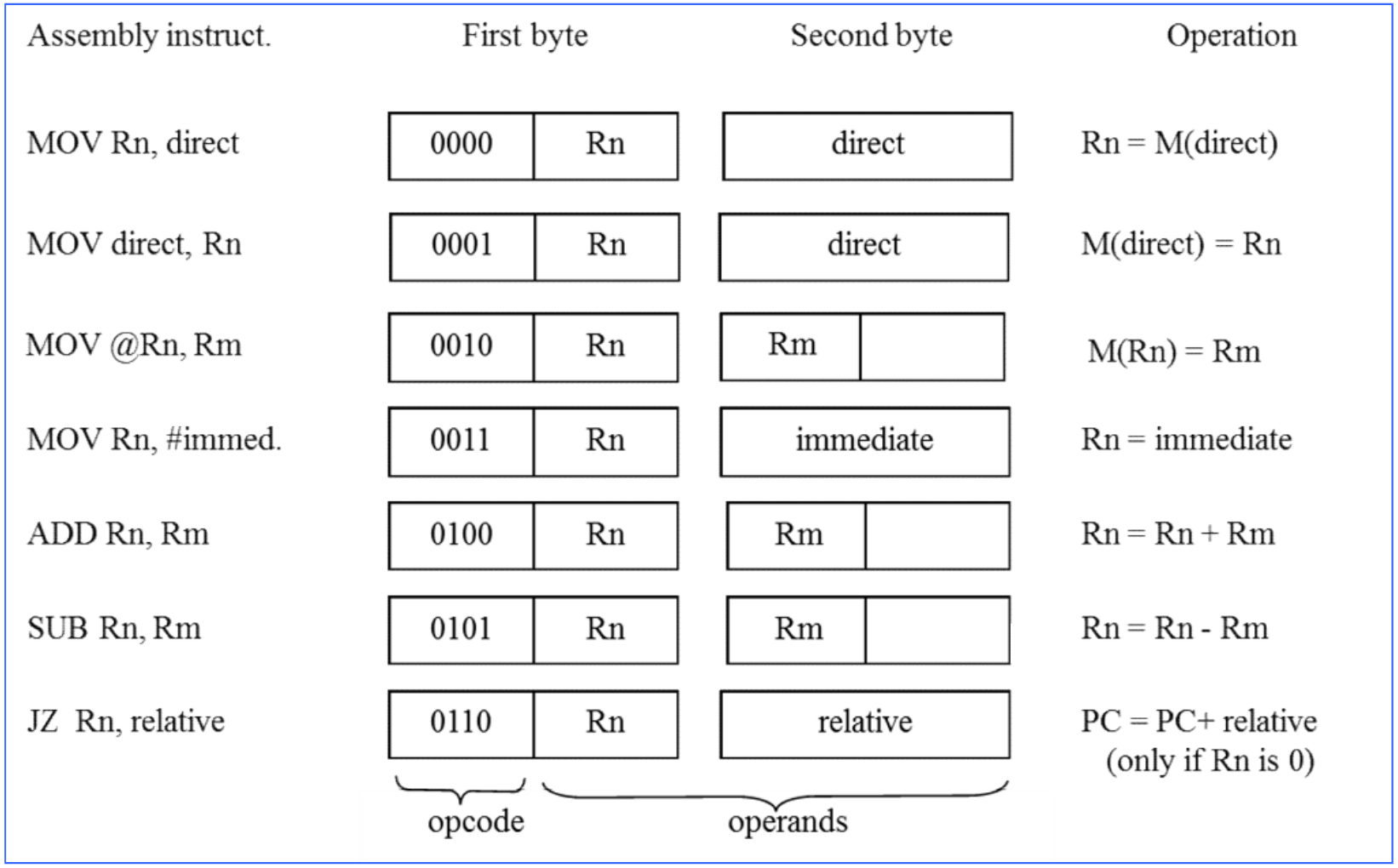
```
        0    MOV R0, #0;        // total = 0
        1    MOV R1, #10;       // i = 10
        2    MOV R2, #1;        // constant 1
        3    MOV R3, #0;        // constant 0

Loop:   JZ R1, Next;            // Done if i=0
        5    ADD R0, R1;        // total += i
        6    SUB R1, R2;        // i--
        7    JZ R3, Loop;       // Jump always
Next:   // next instructions...
```



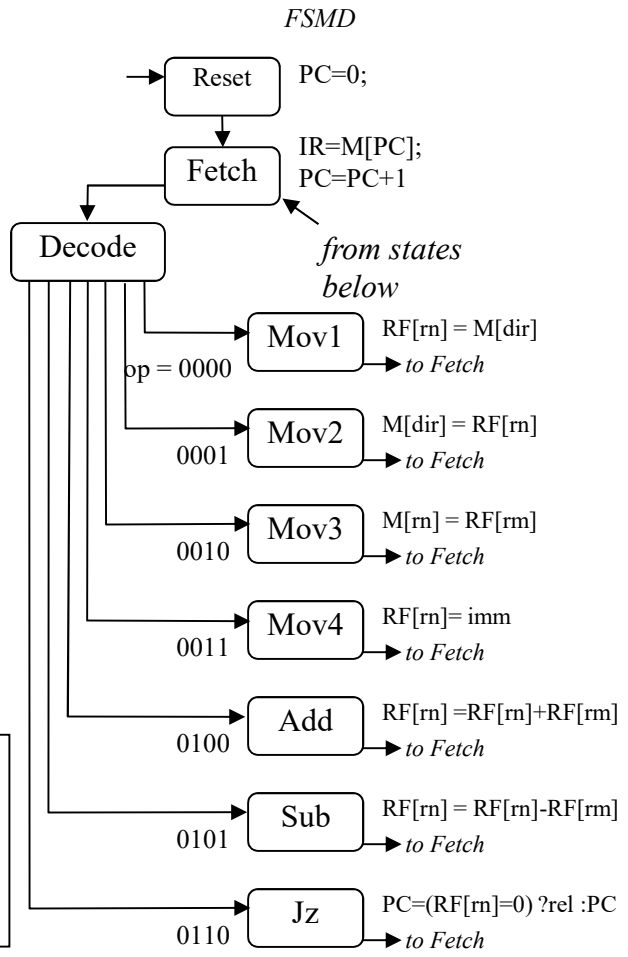
Σχεδιασμός επεξεργαστή Γενικού σκοπού

Άσκηση: Υλοποιήστε σε FSMD τον επεξεργαστή που υλοποιεί το σύνολο εντολών



Σχεδιασμός απλού επεξεργαστή

Assembly instruct.	First byte		Second byte	Operation
MOV Rn, direct	0000	Rn	direct	Rn = M(direct)
MOV direct, Rn	0001	Rn	direct	M(direct) = Rn
MOV @Rn, Rm	0010	Rn	Rm	M(Rn) = Rm
MOV Rn, #immed.	0011	Rn	immediate	Rn = immediate
ADD Rn, Rm	0100	Rn	Rm	Rn = Rn + Rm
SUB Rn, Rm	0101	Rn	Rm	Rn = Rn - Rm
JZ Rn, relative	0110	Rn	relative	PC = PC+ relative (only if Rn is 0)
	opcode		operands	



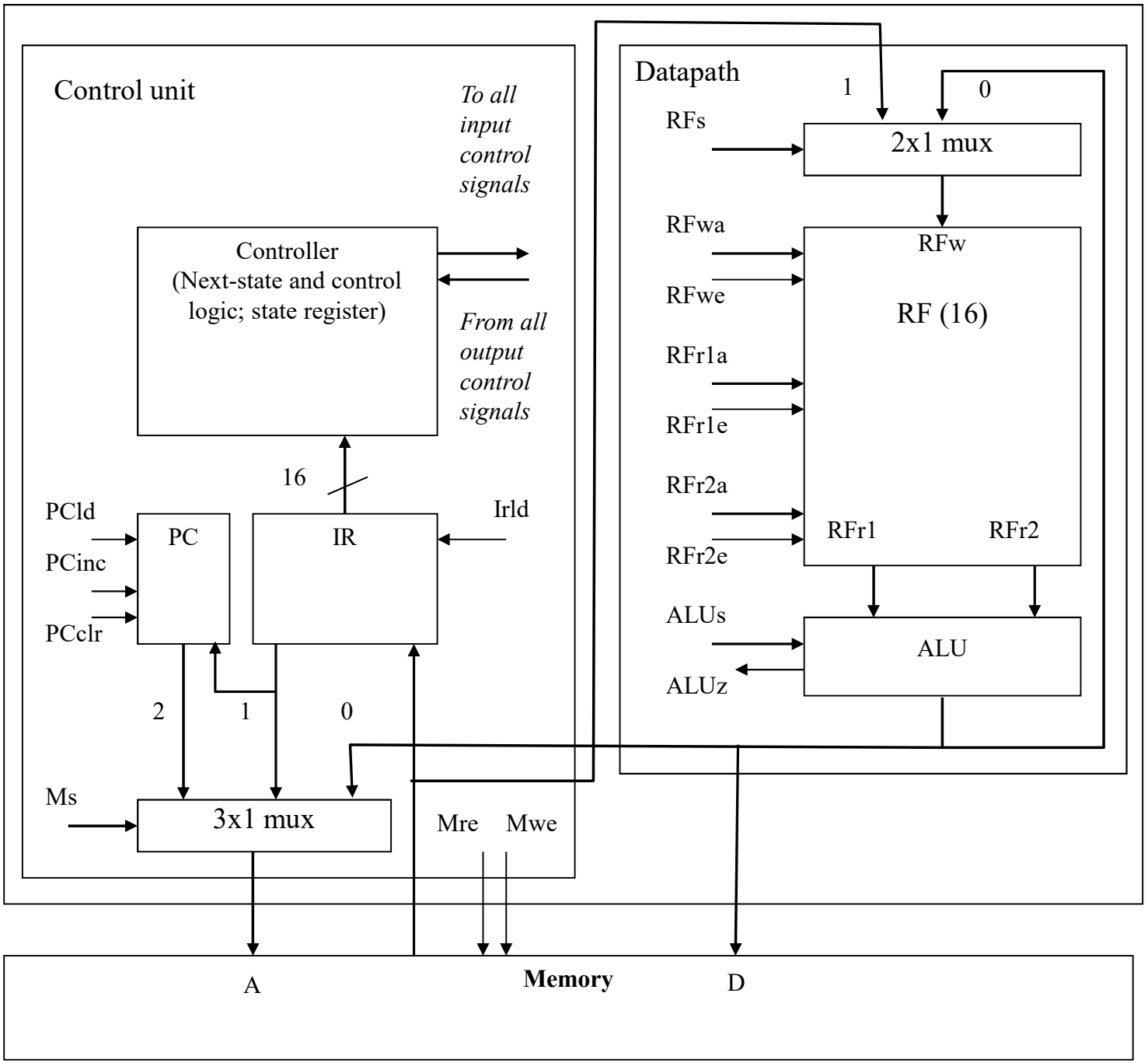
Declarations:

```
bit PC[16], IR[16];
bit M[64k][16], RF[16][16];
```

Aliases:

op	IR[15..12]	dir	IR[7..0]
rn	IR[11..8]	imm	IR[7..0]
rm	IR[7..4]	rel	IR[7..0]

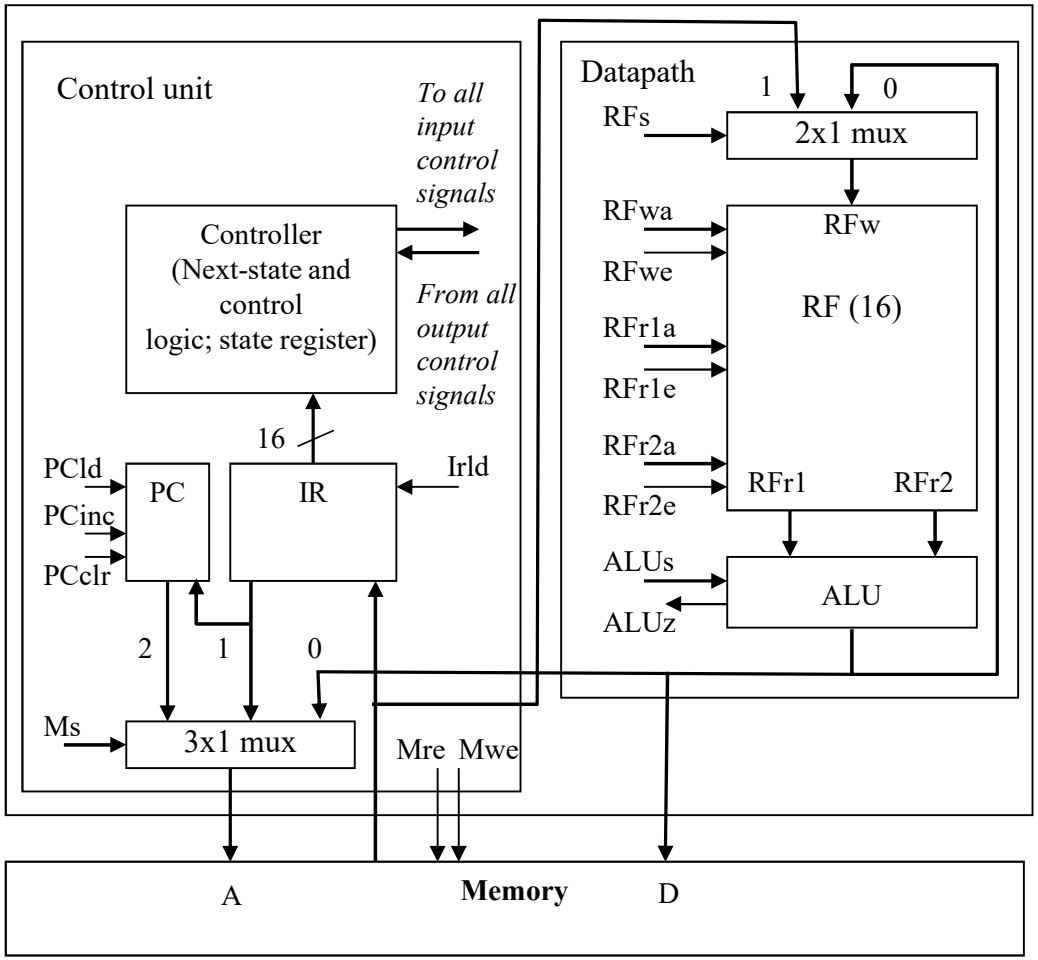
Απλός Επεξεργαστής



Απλός Επεξεργαστής

Q1: Από πόσα bits (πρέπει να) αποτελούνται τα ακόλουθα σήματα:

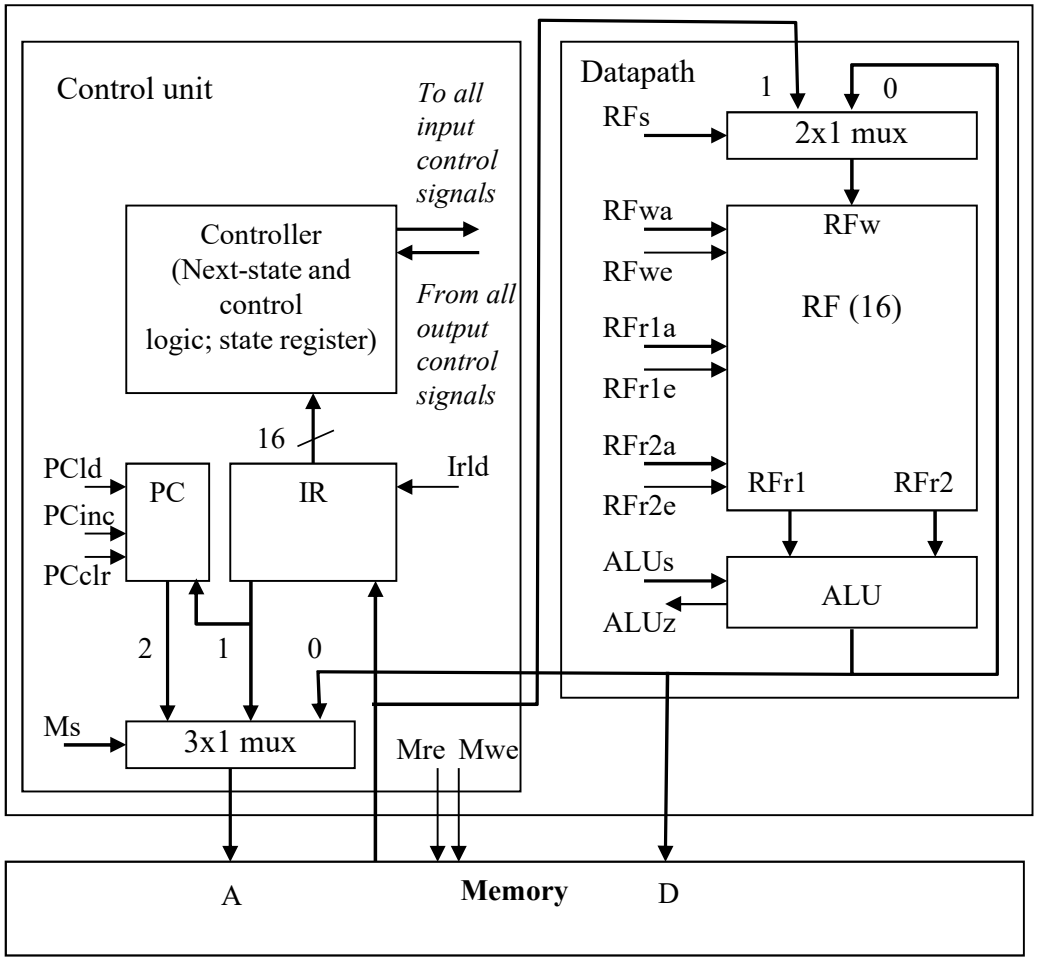
RFwa	PCld
RFwe	PCinc
RFr1a	PCclr
RFr1e	Ms
RFr2a	Irld
RFr2e	Mre
ALUs	Mwe
ALUz	A
D	2
RFw	1
RFs	0



Απλός Επεξεργαστής

Q2: Ποια σήματα πρέπει να ενεργοποιηθούν για να:

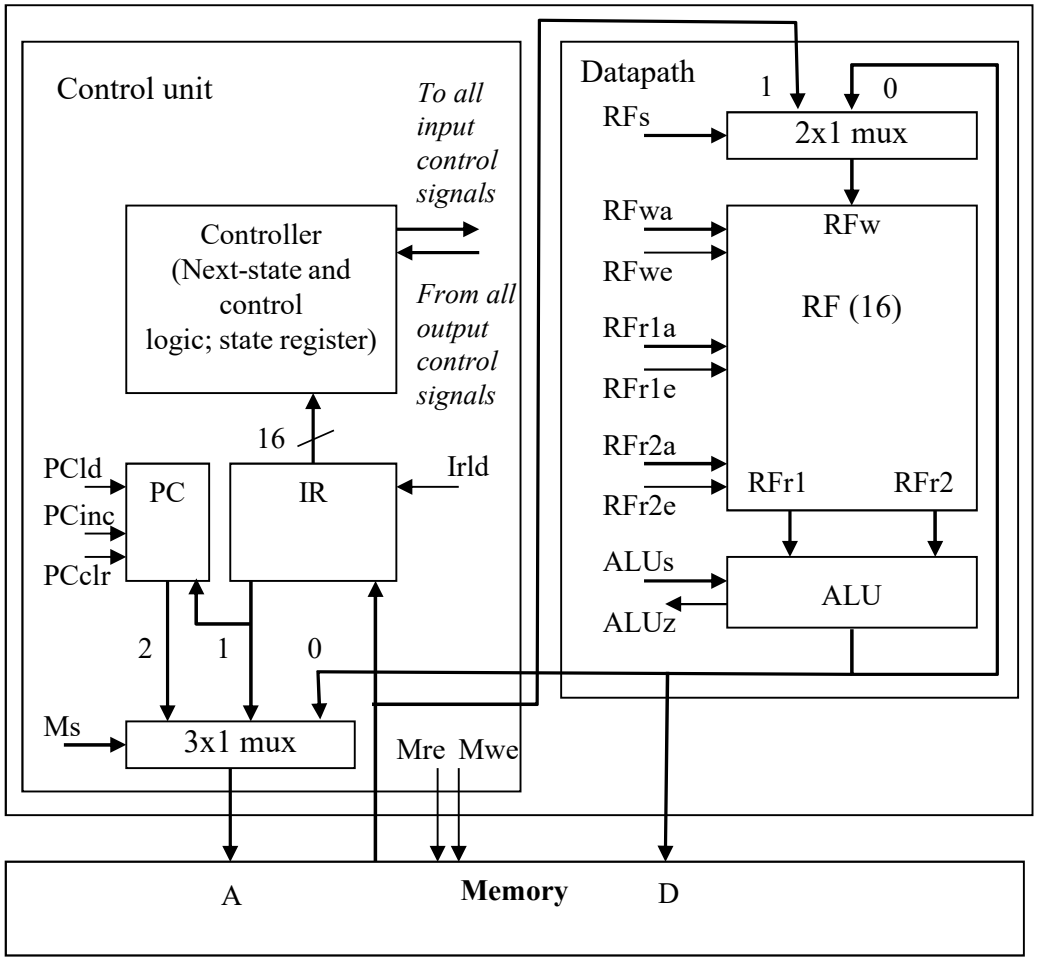
- ALU => στο A
- ALU => στο D
- Διαβάσουμε τον R3
- Γράψουμε στον R3
- Αυξήσουμε τον PC
- PC => A
- IR => A
- IR => PC
- Φορτώσουμε τον IR



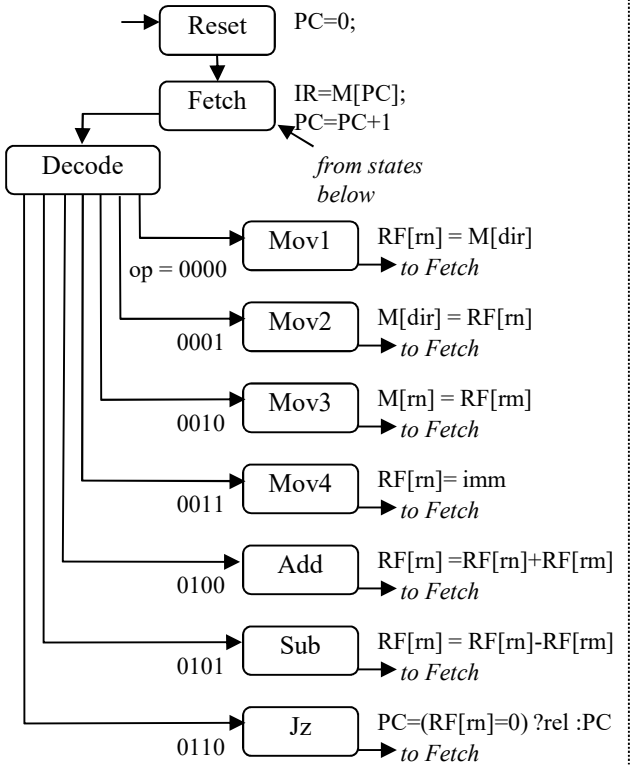
Απλός Επεξεργαστής

Q3: TI πρέπει να γίνει σε κάθε μία από τις ακόλουθες εντολές:

Assembly instruct.	Operation
MOV Rn, direct	Rn = M(direct)
MOV direct, Rn	M(direct) = Rn
MOV @Rn, Rm	M(Rn) = Rm
MOV Rn, #immed.	Rn = immediate
ADD Rn, Rm	Rn = Rn + Rm
SUB Rn, Rm	Rn = Rn - Rm
JZ Rn, relative	PC = relative (only if Rn is 0)



A Simple Microprocessor



FSMD

PCclr=1;

MS=10;
Irld=1;
Mre=1;
PCinc=1;

RFwa=rn; RFwe=1; RFs=01;
Ms=01; Mre=1;

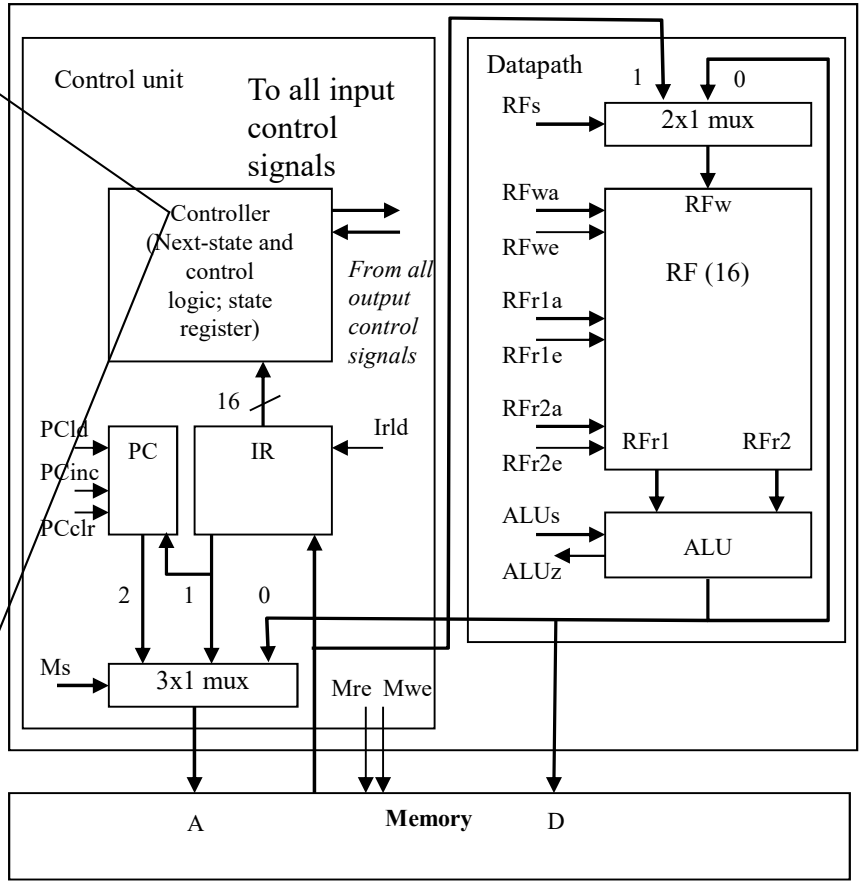
RFr1a=rn; RFr1e=1;
Ms=01; Mwe=1;

RFr1a=rn; RFr1e=1;
Ms=10; Mwe=1;

RFwa=rn; RFwe=1; RFs=10;

RFwa=rn; RFwe=1; RFs=00;
RFr1a=rn; RFr1e=1;
RFr2a=rm; RFr2e=1; ALUs=00
RFwa=rn; RFwe=1; RFs=00;
RFr1a=rn; RFr1e=1;
RFr2a=rm; RFr2e=1; ALUs=01
PCld= ALUz;
RFrla=rn;
RFrle=1;

FSM operations that replace the FSMD operations after a datapath is created



Ένας ΠΟΛΥ απλός επεξεργαστής!

Summary

- General-purpose processors
  - Good performance, flexible
- Controller, datapath, and memory
- Structured languages prevail
  - But some assembly level programming still necessary
- Many tools available
  - Including instruction-set simulators, and in-circuit emulators
- ASIPs
  - Microcontrollers, DSPs, network processors, more customized ASIPs
- Choosing among processors is an important step
- Designing a general-purpose processor is conceptually the same as designing a single-purpose processor