Θέμα 3ο:

	15	14	13	12	11		0
PC:	*instruction1* code				x	. . .	x
PC+1:	*instruction2* code				x	. . .	x
PC+2:	*instruction1* code				x	. . .	x
PC+3:	*instruction2* code				x	. . .	x

Α) Η φάση ανάκλησης (Φ.Α) πρέπει να φροντίζει να διαλέγει μεταξύ των δύο απλών εντολών (instruction1 ή instruction2), ανάλογα με το p. Ο έλεγχος του p γίνεται στην αρχή της Φ.Α.

Εισάγουμε δύο νέες μικρολειτουργίες μ_{24}, μ_{25}:

μ_{24}: Αν p=1 τότε τίποτε
αλλιώς αν p=0 τότε PC=PC+1;

μ_{25}: Αν p=1 τότε PC PC=PC+1
αλλιώς, αν p=0 τότε τίποτα

Οι μ_1, μ_2 και μ_3 διατηρούνται αναλλοίωτες.

Η ΦΑ της σύνθετης εντολής γίνεται (ο χρονισμός αγνοείται) ΦΑ: μ_{24}, μ_1, μ_2, μ_3, μ_{25}

Η ΦΕ της σύνθετης εντολής είναι η ΦΕ της απλής instruction1 ή instruction2.

Σχόλιο:
Στην έναρξη κάθε σύνθετης εντολής, το PC δείχνει πάντα στην πρώτη (instruction1) από τις δύο απλές εντολές.

- Αν το p=1, τότε, η μ_{24} δεν κάνει τίποτε και οι μ_1, μ_2, μ_3 ανακαλούν σωστά την instruction1 από τη μνήμη. Η μ_{25} προχωρά τον PC μια θέση ακόμα + μία θέση που τον προχώρησε η μ_3 άρα, τελικά το PC δείχνει στο PC+2, έτοιμο για την ΦΑ της επόμενης σύνθετης εντολής.
- Αν το p=0, τότε, η μ_{24} αυξάνει το PC κατά 1 και μ_1, μ_2, μ_3 ανακαλούν σωστά την instruction2 από τη μνήμη. Η μ_{25} δεν κάνει τίποτε, άρα το PC δείχνει στην επόμενη θέση που τον προχώρησε η μ_3 άρα, τελικά το PC δείχνει στο PC+1, έτοιμο για την ΦΑ της επόμενης σύνθετης εντολής.

Το p ενδεχομένως να αλλάξει κατά τη φάση εκτέλεσης. Η αλλαγή αυτή μας επηρεάζει στην αρχή (Φ.Α) της επόμενης σύνθετης εντολής σύμφωνα με τα παραπάνω.

Εναλλακτικά (παραλλαγή της παραπάνω λύσης)
Μετατρέπουμε το 1-bit p σε 11 bit register p (γεμίζουμε τα 10 MSbits με 0), βάζουμε αθροιστή στον PC και:
Αλλάζουμε την μ_1 σε μ_1: (MAR) := (PC) + p*, όπου p* είναι το NOT(p)
Όμοια αλλάζουμε την μ_3 σε μ_3: (PC) := (PC) + 1 + p. Η ΦΑ της σύνθετης εντολής εξακολουθεί να είναι μ_1, μ_2, μ_3!

Β) Έστω το παρακάτω απλό πρόγραμμα στην **νέα** assembly του ΕΚΥ:

```
        p ? LDA NUM :  NOP
RSL     p ? STA A1 :  LDA ZERO
        p ? STOP :  SBA NUM
        p ? JMP RSL : NOP
```

Αρχικά, μας λέει η εκφώνηση ότι p=1. Επομένως, φορτώνεται το (NUM) στον Α, όπου (NUM) $\neq 0$.

- Αν το (NUM) >0 τότε αποθηκεύεται στο Α1 και STOP
- Αν το (NUM)<0 τότε, φορτώνεται το (ZERO) στον Α, και επειδή το p εξακολουθεί να είναι 0, αφαιρείται από τον Α το (NUM), άρα ο Α έχει το –(NUM) που είναι πια θετικό, άρα p=1. Επομένως, κατόπιν εκτελείται η JMP RSL και αποθηκεύεται το –(NUM) στο Α1.

Άρα το παραπάνω πρόγραμμα αποθηκεύει στη δ/νση Α1 το |(NUM)|, δηλ. την απόλυτη τιμή του περιεχομένου της δ/νσης NUM.

Β2) Ισοδύναμο πρόγραμμα στην assembly του ΕΚΥ (**προσοχή** ζητείται το ακριβές ισοδύναμο!)

```
        LDA NUM
        JAN NEG
RSL     STA A1
        STOP
NEG     LDA ZERO
        SBA NUM
        JMP RSL
        END
```