ΠΑΝΕΠΙΣΤΗΜΙΟ
ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

Αρχιτεκτονική Υπολογιστών

Ασκήσεις Εργαστηρίου

Ενότητα: ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ No 08

Δρ. Μηνάς Δασυγένης

mdasyg@ieee.org

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Εργαστήριο Ρομποτικής, Ενσωματωμένων και Ολοκληρωμένων Συστημάτων

http://arch.ece.uowm.gr/mdasyg

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ψηφιακά Μαθήματα του Πανεπιστημίου Δυτικής Μακεδονίας**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Περιεχόμενα

1. Σκοπός της άσκησης

- Είσοδος/Έξοδος με Polling.
- Χειρισμός περιφερειακής συσκευής με εντολές εισόδου εξόδου.

(A) 10 Ασκήσεις
(C) 6 Εργασίες
(B) 2 Bonus

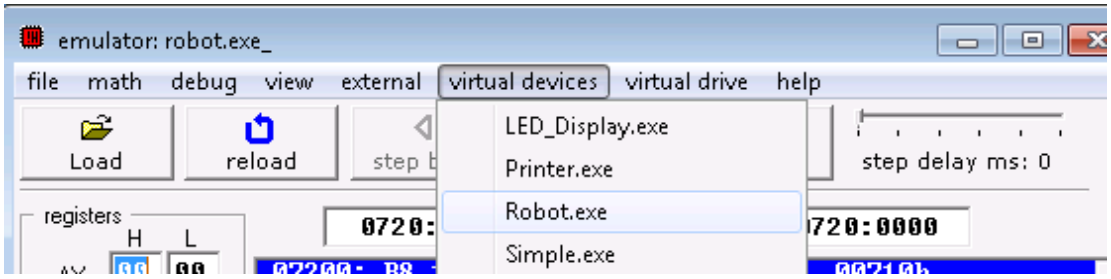
2. Εργαστηριακή Άσκηση

Σε αυτό το εργαστήριο θα αναπτυχθεί ένα πρόγραμμα για το χειρισμό μιας συσκευής η οποία είναι συνδεδεμένη στο σύστημά μας.

Η συσκευή είναι ένα τηλεκατευθυνόμενο όχημα robot το οποίο εκτελεί πλήθος κινήσεων *(μπροστά, στροφή 90ο δεξιά, στροφή 90ο αριστερά)* και ενεργειών *(άνοιγμα λάμπας, κλείσιμο λάμπας, ανίχνευση μπροστινού αντικειμένου).*

Το όχημα είναι μια μηχανική συσκευή και απαιτείται κάποιο χρονικό διάστημα προκειμένου να εκτελέσει μια κατάσταση και να είναι σε θέση ετοιμότητας να δεχτεί μια επόμενη εντολή. Θα πρέπει λοιπόν πριν στείλουμε μια οδηγία να ελέγξουμε την ετοιμότητά της συσκευής *(ότι είναι έτοιμη να δεχτεί μια εντολή).*

→ Για να εμφανίσουμε το όχημα θα πρέπει από το μενού του emulator με όνομα virtual devices, να επιλέξουμε το robot.exe.



Το όχημα επικοινωνεί με τον υπολογιστή μας χρησιμοποιώντας 3 θύρες εισόδου/εξόδου.

- Τη θύρα 9 που είναι θύρα εντολών, δηλαδή στέλνουμε εντολές στο robot.
- Τη θύρα 10 που είναι θύρα δεδομένων, δηλαδή το robot μας στέλνει δεδομένα.
- Τη θύρα 11 που είναι θύρα κατάστασης, δηλαδή το robot μας λέει σε ποια κατάσταση είναι *(αν υπάρχει σφάλμα ή όχι).*

3. Πίνακες εντολών

Οι παρακάτω πίνακες μας δείχνουν τις εντολές που δέχεται και τι δεδομένα επιστρέφει στις θύρες 9,10,11:

Θύρα εντολών (9):

decimal value	binary value	action
0	00000000	do nothing.
1	00000001	move forward.
2	00000010	turn left.
3	00000011	turn right.
4	00000100	examine. examines an object in front using sensor. when robot completes the task, result is set to **data register** and **bit #0** of **status register** is set to **1**.
5	00000101	switch on a lamp.
6	00000110	switch off a lamp.

Θύρα δεδομένων (10):

decimal value	binary value	meaning
255	11111111	wall
0	00000000	nothing
7	00000111	switched-on lamp
8	00001000	switched-off lamp

Θύρα κατάστασης (11):

bit number	description
bit #0	**zero** when there is no new data in **data register**, **one** when there is new data in **data register**.
bit #1	**zero** when robot is ready for next command, **one** when robot is busy doing some task.
bit #2	**zero** when there is no error on last command execution, **one** when there is an error on command execution (when robot cannot complete the task: move, turn, examine, switch on/off lamp).

1. Ξεκινήστε από το Template (B).

2. Πριν από τον τερματισμό του προγράμματος τοποθετήστε την αντίστοιχη εντολή JMP ώστε να πηγαίνει η εκτέλεση πάλι από την αρχή του κώδικα. Με αυτόν τον τρόπο δημιουργούμε ένα αέναο βρόχο που θα εκτελείται συνεχώς *(δε θα τερματίζει ποτέ).* Όλες οι κλήσεις των συναρτήσεων θα βρίσκονται μέσα σε αυτό το βρόχο.

ΠΡΟΣΟΧΗ: Όλες οι συναρτήσεις σας να είναι διαφανείς. Δηλαδή, όποιοι καταχωρητές τροποποιούνται μέσα στις συναρτήσεις σας να έχουν τις αντίστοιχες εντολές PUSH, POP. Με αυτόν τον τρόπο δε θα αλλάζουν τιμή *(δε θα τροποποιείται δηλαδή το περιβάλλον κλήσης)* μετά την επιστροφή από τη συνάρτηση.

→ (Αν δε γίνει αυτό θα υπάρχουν σφάλματα που δε θα μπορείτε να βρείτε με εύκολο τρόπο)

4. Συναρτήσεις

4.1 check_ready

(C1) Δημιουργία της συνάρτησης check_ready

3. Η συνάρτηση check_ready είναι η συνάρτηση η οποία όταν καλείται θα διαβάζει συνεχώς τη θύρα κατάστασης (δηλαδή θα διαβάζει από τη θύρα 11) και θα εξετάζει το bit#1. Αν αυτό το bit είναι 1, τότε θα επαναλαμβάνεται η διαδικασία της ανάγνωσης της θύρας 11 και εξέτασης του bit. Όταν αυτό το bit είναι 0, τότε θα εκτελείται το RET και θα επιστρέφει η συνάρτηση. Αυτού του είδους ή είσοδος/έξοδος ονομάζεται polling. Δημιουργήστε τη συνάρτηση check_ready με τις παρακάτω οδηγίες.
4. Μέσα στη συνάρτηση θα υπάρχει μια ετικέτα **againcheck:**
5. Αμέσως μετά θα διαβάζουμε από τη θύρα 11 μια τιμή byte και θα την τοποθετούμε σε έναν καταχωρητή *(μεγέθους Byte).* Η ανάγνωση της τιμής γίνεται με την εντολή `in al,11.`
6. Μας ενδιαφέρει να εξετάσουμε το bit#1, δηλαδή το δεύτερο bit. Για να απομονώσουμε αυτό το bit θα χρησιμοποιήσουμε τη μάσκα 00000010b και την λογική πράξη AND. Η εντολή γίνεται λοιπόν:

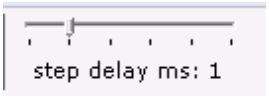
   ```
   and al,00000010b.
   ```

7. Ο al αμέσως μετά την εκτέλεση θα έχει το bit που μας ενδιαφέρει. Αυτό το bit είτε θα έχει τιμή 0, οπότε ο al θα είναι 00000000 δηλαδή θα έχει τιμή 0 (έχουμε απομονώσει αυτό το bit, όλα τα άλλα bit είναι σίγουρα 0), είτε αν το bit#1 έχει τιμή 1, ο al θα έχει τιμή 00000010 δηλαδή θα έχει τιμή 2. Δημιουργήστε τη σύγκριση: **“Αν ο AL έχει τιμή 2, τότε πήγαινε στην ετικέτα againcheck διαφορετικά συνέχισε κανονικά στο RET”.** Με τον παραπάνω βρόχο ο επεξεργαστής δε προχωράει παρακάτω, δηλαδή δεν επιστρέφει από τη συνάρτηση, μέχρι να είναι έτοιμη η συσκευή για επικοινωνία.

8. Τοποθετήστε τις αντίστοιχες εντολές PUSH στην αρχή της συνάρτησης και POP πριν από το RET για τους καταχωρητές που τροποποιούνται *(στο παράδειγμά μας μόνο για τον AX).*

9. Στο κυρίως πρόγραμμα τοποθετήστε την κλήση της συνάρτησης και επιβεβαιώστε την ορθή λειτουργία. Αν το εκτελέσετε δε θα πρέπει να δείτε να κάνει κάτι το robot.

<u>ΠΡΟΣΟΧΗ:</u> Αν έχετε πρόβλημα χρήσης του περιφερειακού, τότε τροποποιήστε το step delay κατά την προσομοίωση.



4.2 print_menu

(C2) Η συνάρτηση print_menu

10. Η συνάρτηση αυτή θα εμφανίζει ένα μενού λειτουργιών στο χρήστη. Το μενού που θα πρέπει να εμφανίζει είναι το παρακάτω:

11. . Μπορείτε να εκτυπώσετε όλο το μενού είτε με πολλαπλές κλήσεις της LEA και θα έχει όνομα το κάθε string παραπάνω, είτε με μια κλήση της LEA στην οποία θα έχετε όλο το string.

12. Τοποθετήστε τις αντίστοιχες εντολές PUSH στην αρχή της συνάρτησης και POP πριν από το RET για τους καταχωρητές που τροποποιούνται .

13. Στο κυρίως πρόγραμμα τοποθετήστε την κλήση της συνάρτησης κάτω από την προηγούμενη συνάρτηση και επιβεβαιώστε την ορθή λειτουργία.

4.3 input_command

(C3) Η συνάρτηση input_command

14. Η συνάρτηση αυτή θα διαβάζει από το πληκτρολόγιο ένα χαρακτήρα χωρίς εμφάνιση. Ο χαρακτήρας θα αντιστοιχεί στο εύρος 1 έως 6, σύμφωνα με το παραπάνω μενού. Όσο ο χρήστης πληκτρολογεί χαρακτήρες που δεν ανήκουν στο εύρος από “1” έως “6” *(προσοχή στην ASCII τιμή)*, τότε θα επαναλαμβάνεται η είσοδος.

15. Μόλις ο χρήστης πληκτρολογήσει μια τιμή από “1” έως “6” τότε η τιμή θα μετατρέπεται στην καθαρή τιμή.

16. Στη συνέχεια η τιμή αυτή θα αποθηκεύεται σε μια θέση μνήμης στο τμήμα δεδομένων με το όνομα command που θα είναι τύπου db.
17. Τοποθετήστε τις αντίστοιχες εντολές PUSH στην αρχή της συνάρτησης και POP πριν από το RET για τους καταχωρητές που τροποποιούνται .
18. Στο κυρίως πρόγραμμα τοποθετήστε την κλήση της συνάρτησης κάτω από την προηγούμενη συνάρτηση και επιβεβαιώστε την ορθή λειτουργία.

4.4 send_command

(C4) Η συνάρτηση send_command

19. Η συνάρτηση αυτή θα διαβάζει την εντολή που έχει αποθηκευτεί στη θέση μνήμης command και θα τη στέλνει στη θύρα εντολών.
20. Κάντε κλήση της check_ready για να βεβαιωθούμε ότι το robot είναι έτοιμο να δεχτεί την εντολή μας.
21. Θα πρέπει να μηδενίσουμε κάθε εντολή που υπάρχει στη θύρα 9. Για να γίνει αυτό θα τοποθετήσετε το 0 στον καταχωρητή al και στη συνέχεια θα στείλετε το al στο robot στη θύρα 9 ως εξής:

    out 9, al

22. Μεταφέρετε στον καταχωρητή al το περιεχόμενο που έχει η θέση μνήμης command.
23. Κάντε κλήση της check_ready για να βεβαιωθούμε ότι το robot είναι έτοιμο να δεχτεί την εντολή μας.
24. Στείλτε την τιμή που έχει ο al στο robot στη θύρα 9.
25. Τοποθετήστε τις αντίστοιχες εντολές PUSH στην αρχή της συνάρτησης και **POP** πριν από το **RET** για τους καταχωρητές που τροποποιούνται .
26. Στο κυρίως πρόγραμμα τοποθετήστε την κλήση της συνάρτησης κάτω από την προηγούμενη συνάρτηση και επιβεβαιώστε την ορθή λειτουργία. Πατήστε τους χαρακτήρες οδήγησης *(1, 2, 3)* και τους χαρακτήρες που ανάβουν ή σβήνουν λάμπα, σύμφωνα με τους παραπάνω πίνακες. Παρατηρήστε ότι το robot σας υπακούει και εκτελεί τις εντολές σας.

4.5 check_command_execution

(C5) Η συνάρτηση check_command_execution

27. Αυτή η συνάρτηση θα εκτελείται αμέσως μετά την αποστολή μιας εντολής στο robot και θα μας αναφέρει αν υπάρχει σφάλμα. Για παράδειγμα αν οδηγήσετε το robot σε ένα τοίχο και προκαλέσετε σύγκρουση τότε θα δημιουργηθεί σφάλμα. Επίσης, αν δώσετε εντολή να σβήσει μια λάμπα που είναι ήδη σβηστή θα δημιουργήσει σφάλμα.
28. Δημιουργήστε ένα string μέσα στο τμήμα δεδομένων με το όνομα errormsg τύπου db και με μήνυμα “ERROR”

29. Μέσα σε αυτή τη συνάρτηση καλέστε τη συνάρτηση check_ready για να βεβαιωθούμε ότι μπορούμε να επικοινωνήσουμε με το robot.
30. Διαβάστε από τη θύρα 11 *(θύρα κατάστασης)* και τοποθετήστε το περιεχόμενο στον καταχωρητή AL. Ο AL τώρα θα έχει όλα τα bit που μας έδωσε η θύρα 11.
31. Σύμφωνα με τον πίνακα για τη θύρα 11 μας ενδιαφέρει το bit#2. Δημιουργήστε την κατάλληλη μάσκα απομόνωσης του bit αυτού και ταυτόχρονα κάντε την ανάλογη ΛΟΓΙΚΗ πράξη.
32. Ο καταχωρητής al ύστερα από την παραπάνω πράξη θα έχει είτε τιμή 00000100 είτε 00000000. Δημιουργήστε τον επόμενο έλεγχο:

    Αν ο AL έχει τιμή 4 τότε να εκτυπωθεί το μήνυμα errormsg, διαφορετικά η προηγούμενη εντολή έχει εκτελεστεί με επιτυχία και δε χρειάζεται να εκτυπωθεί κάτι.

33. Τοποθετήστε τις αντίστοιχες εντολές PUSH στην αρχή της συνάρτησης και POP πριν από το RET για τους καταχωρητές που τροποποιούνται .
34. Η συνάρτηση αυτή θα καλείται μέσα από τη συνάρτηση send_command ακριβώς κάτω από τη γραμμή out 9,al στην οποία στέλνουμε την εντολή προς το robot.
35. Επιβεβαιώστε την ορθή λειτουργία με το να οδηγήσετε το robot σε μια σύγκρουση με ένα τοίχο είτε να σβήσετε μια σβηστή λάμπα. Θα πρέπει να σας εμφανιστεί το μήνυμα “ERROR”.

4.6 examine

(C6) Η συνάρτηση examine

36. Η συνάρτηση αυτή θα καλείται όταν ο χρήστης πατήσει το “4”. Η συνάρτηση αυτή θα λέει στο χρήστη αν το αντικείμενο μπροστά στο robot είναι τοίχος, κλειστή λάμπα ή ανοιχτή λάμπα. Αν δεν υπάρχει αντικείμενο μπροστά δε θα εμφανίζει τίποτα.
37. Τοποθετήστε στο τμήμα δεδομένων τα string:

    1. **wallmsg**, το οποίο θα έχει το μήνυμα “Wall”
    2. **offlampmsg**, το οποίο θα έχει το μήνυμα “switched-off lamp”
    3. **onlampmsg**, το οποίο θα έχει το μήνυμα “switched-on lamp”

38. Μέσα στη συνάρτηση αυτή καλέστε τη συνάρτηση check_ready
39. Διαβάστε από τη θύρα *10 (θύρα δεδομένων)* και τοποθετήστε την τιμή στον καταχωρητή **AL**.
40. Ο **AL** θα έχει μια από τις τέσσερις αποδεκτές τιμές που εμφανίζονται στον πίνακα στην αρχή του εργαστηρίου. Υλοποιήστε τους παρακάτω ελέγχους:

    1. Αν η τιμή του AL είναι 255, τότε να εμφανιστεί το μήνυμα **wallmsg**
    2. Αν η τιμή του AL είναι 7, τότε να εμφανιστεί το μήνυμα **onlampmsg**
    3. Αν η τιμή του AL είναι 8, τότε να εμφανιστεί το μήνυμα **offlampmsg**
    4. Αν ο AL έχει τιμή 0 δε θα εμφανίζεται **τίποτα**.

41. Τοποθετήστε τις αντίστοιχες εντολές **PUSH** στην αρχή της συνάρτησης και **POP** πριν από το **RET** για τους καταχωρητές που τροποποιούνται .
42. Η συνάρτηση αυτή θα καλείται μέσα από τη συνάρτηση send_command αμέσως μετά την κλήση της συνάρτησης check_command_execution, Αν ο χρήστης έχει ζητήσει τη λειτουργία examine, δηλαδή Αν το **AL** έχει την τιμή 4.
43. Επιβεβαιώστε την ορθή λειτουργία με το να οδηγήσετε το robot σε μπροστά από αντικείμενα και να πατάτε το πλήκτρο examine. Θα πρέπει να σας αναφέρει το είδος του αντικειμένου.

5. Ερωτήσεις Κατανόησης

1. **(A1)** Σε μια εξωτερική μνήμη, υπάρχουν τα εξής δεδομένα: Στο BYTE0 υπάρχει η τιμή FFh ενώ στο BYTE1 υπάρχει η τιμή 01h. Αν αυτά τα 2Byte αντιπροσωπεύουν μια τιμή συμπληρώματος ως προς 2 ποια είναι η δεκαδική προσημασμένη τιμή ΑΝ αυτά τα 2 Byte αντιστοιχούν σε:

    1. οργάνωση big-endian
    2. οργάνωση little-endian

2. **(A2)** Ποια είναι τα 3 προβλήματα στο παρακάτω κομμάτι κώδικα;

```
function1 proc
        cmp al,4
        jne next_iteration
        call examine

        POP AX
        RET
function1 endp

function3 proc
        ush dx
        ush ax
next_iteration:
        lea dx,offlampmsg
        mov ah,09
        int 21h
        pop ax
        pop dx
        ret
function3 endp
```

3. **(A3)** Ποια είναι τα 3 συντακτικά προβλήματα στο παρακάτω κομμάτι κώδικα;

```
function1 proc
        push al
                mov al,4

                call examine
                out 9,0 ;steile to 0 sth thira 9
                POP Al
                RET
function2 endp
```

4. **(A4)** Γράψτε 3 διαφορετικούς τρόπους για να κάνουμε τη λογική πράξη AND με την τιμή 00000010b.
5. **(A5)** Στο πρόγραμμά σας ποια είναι η μέγιστη χρήση του σωρού σε Byte;
6. **(A6)** Αποκωδικοποιήστε την παρακάτω εντολή βήμα-προς-βήμα *(αναλυτική παρουσίαση)* E8h 63h 00h.
7. **(A7)** Αποκωδικοποιήστε την παρακάτω εντολή βήμα-προς-βήμα *(αναλυτική παρουσίαση)* EBh F2h.
8. **(A8)** Αποκωδικοποιήστε την παρακάτω εντολή βήμα-προς-βήμα *(αναλυτική παρουσίαση)* E6h 09h.
9. **(A9)** Αποκωδικοποιήστε την παρακάτω εντολή 58h.
10. **(A10)** Πόσα Bytes είναι το τμήμα κώδικα και πόσα Bytes το τμήμα δεδομένων του προγράμματός σας; Πως το υπολογίσατε;

6. BONUS

6.1 BONUS1 Αυτοματοποίησης

(+100% μονάδες αν υλοποιηθεί το παρακάτω)

Η παρακάτω συνάρτηση random_command δίνει μια τυχαία τιμή από 0 έως και 3 στη θέση μνήμης command.

```
random_command proc
        PUSH AX
        PUSH DX
        PUSH CX
        mov AH,0        ;Read ticks.
        int 1Ah         ;Time of day interrupt.
                        ;To DX low word
                        ;To CX high word
        and DX,11b
        mov command, dl
        POP CX
        POP DX
        POP AX
        RET
random_command endp
```

Να αυτοματοποιήσετε το πρόγραμμά σας, ώστε να ψάχνει και να βρίσκει τη σβηστή λάμπα και να την ανάβει. Μερικές Οδηγίες:

- Να τοποθετήσετε μια νέα καταχώρηση στο μενού με τίτλο **7 Autopilot**
- Να τροποποιήσετε κατάλληλα το input_command ώστε να δεχόμαστε και την τιμή 7 ως έγκυρη τιμή και να την τοποθετεί στο command.
- Να τροποποιήσετε το κυρίως πρόγραμμα, ώστε:
    - Αν ο χρήστης πατήσει 7 τότε ΔΕ θα καλείται η συνάρτηση **send_command**, αλλά η συνάρτηση **autopilot**.
    - Αν ο χρήστης πατήσει οποιοδήποτε άλλο πλήκτρο θα καλείται η **send_command** κανονικά.

- Η συνάρτηση autopilot θα κάνει το εξής:
    - θα προχωράει συνεχώς μπροστά μέχρι να συγκρουστεί με κάποιο αντικείμενο *(δηλαδή στη θύρα status θα υπάρχει τιμή στο bit#2)*
    - Αν συγκρουστεί
        - θα καλεί τη συνάρτηση examine για να δει αν είναι λάμπα. Αν είναι λάμπα ΚΑΙ είναι σβηστή θα την ανάβει, διαφορετικά δε θα την ανάβει.
        - Θα καλεί τη συνάρτηση random_command για να δημιουργηθεί μια νέα εντολή.
        - Θα εκτελείται η νέα εντολή.
    - Θα πηγαίνει στο βήμα **(α)** δηλαδή στο βήμα να προχωράει συνεχώς μπροστά μέχρι να συγκρουστεί. Αυτό θα επαναλαμβάνεται συνεχώς.

6.2 BONUS2 Ευφυΐας

(2 μονάδες αν υλοποιηθεί το παρακάτω και εξεταστεί σε αυτό ο φοιτητής)

- Τοποθετήστε ευφυΐα στο πρόγραμμά σας, ώστε να μπορεί να μετακινηθεί προς τη λάμπα με έξυπνο τρόπο. Αν υλοποιήσετε το προηγούμενο βήμα θα δείτε ότι το όχημα τις περισσότερες φορές ακολουθεί μια μη έξυπνη διαδρομή ιδιαίτερα αν υπάρχουν πολλά εμπόδια *(κολλάει σε μια περιοχή)*.

- Θα πρέπει να υλοποιηθεί κάποιος αλγόριθμος που να δημιουργεί ένα έξυπνο όχημα το οποίο θα μπορεί π.χ. Να καταγράφει τις διαδρομές που έχει ακολουθήσει είτε να μπορεί με κάποιο τρόπο να μην επαναλαμβάνει διαδρομές που έχει κάνει, ώστε κάποια στιγμή να πλησιάζει τη λάμπα και να τη σβήνει.

- Για διευκόλυνση σας μπορείτε να ζητήσετε από το χρήστη να σας δώσει τον αριθμό γραμμής και στήλης που βρίσκεται εκείνη τη στιγμή το robot, όπως και την κατεύθυνση που έχει (κάτω, πάνω, δεξιά ή αριστερά). Δε μπορείτε να δώσετε τίποτα άλλο.

- Δοκιμάστε το όχημα στους ενδεικτικούς παρακάτω χάρτες *(μπορείτε να δημιουργήσετε το χάρτη αν πατήσετε στα εικονίδια στο κάτω μέρος του παραθύρου αυτού και στη συνέχεια να κάνετε κλικ πάνω στο παράθυρο του χώρου):*

- Αν το πρόγραμμα που έχετε κατασκευάσει μπορεί να βρει τη κλειστή λάμπα μέσα σε 10 λεπτά στο emu8086 σε οποιοδήποτε χάρτη, τότε θα πάρετε το bonus.

- Για διευκόλυνση σας θεωρείτε ότι το robot πάντα ξεκινάει από την πάνω αριστερή γωνία.