Ο Εκπαιδευτικός Υπολογιστής TRN+

1. Δομή του TRN +

Η δομή του TRN+ είναι ελαφρώς τροποποιημένη σε σχέση με αυτή του TRN, επειδή έχουν προστεθεί / τροποποιηθεί ορισμένες λειτουργίες. Ο επεξεργαστής περιλαμβάνει:

- 1 καταχωρητή Buffer Register (BR) των 20bit
- 1 καταχωρητή Διευθύνσεων (AR) των 13bit
- 1 καταχωρητή Συσσωρευτή (A) των 20bit
- 1 βοηθητικό Καταχωρητή (X) των 20bit
- 1 καταχωρητή Δείκτη (I) των 13bit
- 1 καταχωρητή Μετρητή Προγράμματος (PC) των 13bit
- 1 καταχωρητή Εντολής (IR) των 20bit
- 1 καταχωρητή Στοίβας (SP) των 13 bit
- Βοηθητικούς Καταχωρητές (V, Z, S, καθώς και H)

Η κατάσταση όλων των καταχωρητών είναι ορατή μέσα από το γραφικό περιβάλλον του προγράμματος εξομοίωσης. Επιπλέον, υπάρχει το ρολόι του επεξεργαστή το οποίο συγχρονίζει τους κύκλους μηχανής του συστήματος.

2. Εντολές του TRN+

Ο TRN+ έχει πεδίο κώδικα εντολής (Instruction Code) των 5bit, με αποτέλεσμα να υποστηρίζει 32 διαφορετικούς κώδικες εντολής. Η έκδοση του TRN+ περιλαμβάνει 42 εντολές, ορισμένες από τις οποίες έχουν κοινό κώδικα εντολής.Υπάρχουν νέες εντολές που αποσκοπούν στο χειρισμό στοίβας και στο χειρισμό των εσωτερικών καταχωρητών (αύξηση, μείωση και ολίσθηση) . Αναλυτικά οι εντολές του TRN+ έχουν ως εξής:

1. **Καμία Λειτουργία (No OPeration)**

   Δεν επιτελείται καμία λειτουργία.

   Κώδικας : 00000

   Μνημονικό : NOP

Εντολές μεταφοράς

2. **Φόρτωσε τον A (LoaD A)**

   Το περιεχόμενο της διεύθυνσης T μεταφέρεται στον καταχωρητή A, δηλαδή A←M[T].

Κώδικας : 00001

Μνημονικό : LDA

3. **Φόρτωσε τον X (LoaD X)**

   Το περιεχόμενο της διεύθυνσης T μεταφέρεται στον καταχωρητή A, δηλαδή X← M[T].

   Κώδικας : 00010

   Μνημονικό : LDX

4. **Φόρτωσε τον I (LoaD I)**

   Το περιεχόμενο της διεύθυνσης T (τα bits 0 έως και 12) μεταφέρεται στον καταχωρητή I, δηλαδή I← M[T].

   Κώδικας : 00011

   Μνημονικό : LDI

5. **Αποθήκευσε τον A (STore A)**

   Το περιεχόμενο του A αποθηκεύεται στη θέση T της μνήμης, δηλαδή M[T]←A.

   Κώδικας : 00100

   Μνημονικό : STA

6. **Αποθήκευσε τον X (STore X)**

   Το περιεχόμενο του X αποθηκεύεται στη θέση T της μνήμης, δηλαδή M[T]←X.

   Κώδικας : 00101

   Μνημονικό : STX

7. **Αποθήκευσε τον I (STore I)**

   Το περιεχόμενο του I αποθηκεύεται στη θέση T της μνήμης με επέκταση προσήμου, δηλαδή M[T]←I.

   Κώδικας : 00110

   Μνημονικό : STI

8. **Προσδιόρισε τον A (ENter A)**

   Το περιεχόμενο του A γίνεται ίσο με τον αριθμό που υπάρχει στο πεδίο διεύθυνσης της εντολής, δηλαδή A←AP με επέκταση προσήμου.

   Κώδικας : 00111

   Μνημονικό : ENA

9. **Προσδιόρισε τον I (ENter I)**

   Το περιεχόμενο του I γίνεται ίσο με τον αριθμό που υπάρχει στο πεδίο διεύθυνσης της εντολής, δηλαδή I←AP με επέκταση προσήμου.

   Κώδικας : 01011

   Μνημονικό : ENI

10. **Αύξησε τον A (INcrease A)**

    Αυξάνεται το περιεχόμενο του A κατά μονάδα, δηλαδή A←A + 1

    Κώδικας : 01010 Επιπλέον Bit Ελέγχου: $\overline{IR(2)}\ \overline{IR(1)}\ \overline{IR(0)}$

    Μνημονικό : INA

11. **Αύξησε τον X (INcrease X)**

    Αυξάνεται το περιεχόμενο του X κατά μονάδα, δηλαδή X←X + 1

    Κώδικας : 01010 Επιπλέον Bit Ελέγχου: $\overline{IR(2)}\ \overline{IR(1)}\ IR(0)$

    Μνημονικό : INX

12. **Αύξησε τον I (INcrease I)**

    Αυξάνεται το περιεχόμενο του I κατά μονάδα, δηλαδή I←I + 1

    Κώδικας : 01010 Επιπλέον Bit Ελέγχου: $\overline{IR(2)}\ IR(1)\ \overline{IR(0)}$

    Μνημονικό : INI

13. **Ελάττωσε τον A (DeCrease A)**

    Ελαττώνεται το περιεχόμενο του A κατά μονάδα, δηλαδή A←A - 1

Κώδικας : 01010 Επιπλέον Bit Ελέγχου: $\overline{IR(2)}\ IR(1)\ IR(0)$

Μνημονικό : DCA

14. **Ελάττωσε τον X (DeCrease X)**

Ελαττώνεται το περιεχόμενο του X κατά μονάδα, δηλαδή X←X - 1

Κώδικας : 01010 Επιπλέον Bit Ελέγχου: $IR(2)\ \overline{IR(1)}\ \overline{IR(0)}$

Μνημονικό : DCX

15. **Ελάττωσε τον I (DeCrease I)**

Ελαττώνεται το περιεχόμενο του I κατά μονάδα, δηλαδή I←I - 1

Κώδικας : 01010 Επιπλέον Bit Ελέγχου: $IR(2)\ \overline{IR(1)}\ IR(0)$

Μνημονικό : DCI

<u>Αριθμητικές – Λογικές Εντολές</u>

16. **Πρόσθεσε στον A (ADa to A)**

Το περιεχόμενο της θέσης μνήμης T προστίθεται στον A και το αποτέλεσμα οδηγείται στον A, δηλαδή A ← A + M[T].

Κώδικας : 01101

Μνημονικό : ADA

17. **Αφαίρεσε από τον A (SUBtract from A)**

Το περιεχόμενο της θέσης T της μνήμης αφαιρείται από τον A και το αποτέλεσμα οδηγείται στον A, δηλαδή A←A – M[T].

Κώδικας : 01110

Μνημονικό : SUB

18. **Λογικό γινόμενο (Logical AND)**

Εκτελείται το λογικό γινόμενο (AND) μεταξύ των αντίστοιχων bits του A και της θέσης μνήμης T, το δε αποτέλεσμα οδηγείται στον A, δηλαδή A ← A ∧ M[T].

Κώδικας : 01111

Μνημονικό : AND

19. **Λογικό άθροισμα (Logical OR)**

Εκτελείται το λογικό άθροισμα (OR) μεταξύ των αντίστοιχων bits του A και της θέσης μνήμης T, το δε αποτέλεσμα οδηγείται στον A, δηλαδή A ← A ∨ M[T].

Κώδικας : 10000

Μνημονικό : ORA

20. **Αποκλειστικό Η (eXclusive OR)**

Εκτελείται το αποκλειστικό Η (OR) μεταξύ των αντίστοιχων bits του A και της θέσης μνήμης T, δηλαδή A ← A ⊕ M[T].

Κώδικας : 10001

Μνημονικό : XOR

21. **Συμπλήρωμα του A (CoMplement of A)**

Εκτελείται το συμπλήρωμα ως προς ένα του A, δηλαδή τα bits με τιμή "1" γίνονται "0" και αντιστρόφως (A ← $\overline{A}$).

Κώδικας : 10010

Μνημονικό : CMA

Εντολές άλματος

22. **Εντολή άλματος χωρίς συνθήκη (JuMP)**

Ο μετρητής προγράμματος γίνεται ίσος με T οπότε η επόμενη εντολή που θα εκτελεστεί είναι η T, δηλαδή PC ← T.

Κώδικας : 10011

Μνημονικό : JMP

23. **Εντολή άλματος εάν A < 0 (JumP if A Negative)**

    Ο μετρητής προγράμματος γίνεται ίσος με T μόνον εάν το περιεχόμενο του A είναι αρνητικό, δηλαδή εάν A < 0, τότε PC ← T.

    Κώδικας : 10100

    Μνημονικό : JPN

24. **Εντολή άλματος εάν A > 0 (JumP if A Greater than 0)**

    Ο μετρητής προγράμματος γίνεται ίσος με T μόνον εάν το περιεχόμενο του A είναι θετικό, δηλαδή εάν A > 0 , τότε PC ← T.

    Κώδικας : 10101

    Μνημονικό : JAG

25. **Εντολή άλματος εάν A = 0 (JumP if A Zero)**

    Ο μετρητής προγράμματος γίνεται ίσος με T μόνον εάν το περιεχόμενο του A είναι μηδέν, δηλαδή εάν A = 0, τότε PC ← T.

    Κώδικας : 10110

    Μνημονικό : JPZ

26. **Εντολή άλματος εάν υπάρχει υπερχείλιση (JumP on Overflow)**

    Ο μετρητής προγράμματος γίνεται ίσος με T μόνον όταν υπάρχει υπερχείλιση. Υπερχείλιση συμβαίνει όταν το αποτέλεσμα μιας αριθμητικής πράξης είναι λάθος. Συμβολικά η εντολή σημαίνει: Εάν V = 1, τότε PC ← T.

    Κώδικας : 10111

    Μνημονικό : JPO

27. **Εντολή άλματος σε υποπρόγραμμα (JumP to SubRoutine)**

    Εκτελείται άλμα σε υπορουτίνα. Αρχικά αυξάνεται ο δείκτης στοίβας SP κατά 1. Στη συνέχεια ο μετρητής προγράμματος σώζεται στη θέση μνήμης που δείχνει ο SP. Τέλος ο μετρητής προγράμματος παίρνει την τιμή του πεδίου διεύθυνσης της εντολής.

    Κώδικας : 11000

Μνημονικό : JSR

28. **Εντολή άλματος εάν I > 0 (JumP if I Greater than 0)**

    Ο μετρητής προγράμματος γίνεται ίσος με T μόνον εάν το περιεχόμενο του καταχωρητή δείκτη είναι μεγαλύτερο του μηδενός, δηλαδή εάν I > 0 τότε PC ← T.

    Κώδικας : 11001

    Μνημονικό : JIG

Εντολές ολίσθησης

29. **Ολίσθηση του A προς τα αριστερά (SHift A Left)**

    Το περιεχόμενο του A ολισθαίνει προς τα αριστερά κατά μία θέση.

    Κώδικας : 11010 Επιπλέον Bit Ελέγχου: $\overline{IR(1)}\ \overline{IR(0)}$

    Μνημονικό : SHAL

30. **Ολίσθηση του A προς τα δεξιά (SHift A Right)**

    Το περιεχόμενο του A ολισθαίνει προς τα δεξιά κατά μία θέση.

    Κώδικας : 11010 Επιπλέον Bit Ελέγχου: $\overline{IR(1)}\ IR(0)$

    Μνημονικό : SHAR

31. **Ολίσθηση του X προς τα αριστερά (SHift X Left)**

    Όπως και η SHAL αλλά τώρα ολισθαίνει ο καταχωρητής X.

    Κώδικας : 11010 Επιπλέον Bit Ελέγχου: $IR(1)\ \overline{IR(0)}$

    Μνημονικό : SHXL

32. **Ολίσθηση του X προς τα δεξιά (Shift X Right)**

    Όπως και η SHAR αλλά τώρα ολισθαίνει ο καταχωρητής X.

Κώδικας : 11010 Επιπλέον Bit Ελέγχου: $IR(1)\ IR(0)$

Μνημονικό : SHXR

33. **Ολίσθηση του Α και Χ (Shift AX Left)**

Όπως και η SHA αλλά τώρα ολισθαίνει ο συνδυασμός των καταχωρητών Α κα Χ, σαν ένας καταχωρητής αριστερά.

Κώδικας : 11100 Επιπλέον Bit Ελέγχου: $\overline{IR(0)}$

Μνημονικό : SAXL

34. **Ολίσθηση του Α και Χ (Shift AX Right)**

Όπως και η SHA αλλά τώρα ολισθαίνει ο συνδυασμός των καταχωρητών Α κα Χ, σαν ένας καταχωρητής δεξιά.

Κώδικας : 11100 Επιπλέον Bit Ελέγχου: $IR(0)$

Μνημονικό : SAXR

Εντολές εισόδου – εξόδου

35. **Εντολή εισόδου (INPut)**

Το περιεχόμενο του καταχωρητή Α γίνεται ίσο με το περιεχόμενο της θύρας εισόδου Τ.

Κώδικας : 11101 Επιπλέον Bit Ελέγχου: $\overline{IR(0)}$

Μνημονικό : INP

36. **Εντολή εξόδου (OUTput)**

Το περιεχόμενο του καταχωρητή Α μεταφέρεται στην θύρα εξόδου Τ.

Κώδικας : 11101 Επιπλέον Bit Ελέγχου: $IR(0)$

Μνημονικό : OUT

Άλλες εντολές

37. **Διακοπή Λειτουργίας (HaLT)**

Διακόπτεται η λειτουργία του TRN+. Η επαναλειτουργία απαιτεί την ενεργοποίηση ειδικού πλήκτρου.

Κώδικας : 11111

Μνημονικό : HLT

Εντολές χειρισμού στοίβας

38. **Αρχικοποίηση δείκτη στοίβας (Load Stack Pointer)**

Ο δείκτης στοίβας παίρνει την τιμή της θέσης μνήμης όπου θα ξεκινάει η στοίβα.

Κώδικας : 01100

Μνημονικό : LSP

39. **Προσθήκη στη στοίβα (PuSH)**

Ο δείκτης στοίβας αυξάνεται κατά 1 και κατόπιν το περιεχόμενο του Α μεταφέρεται στη στοίβα, δηλαδή στη θέση μνήμης όπου δείχνει ο SP.

Κώδικας : 01000

Μνημονικό : PSH

40. **Εξαγωγή από τη στοίβα (POP)**

Το περιεχόμενο της διεύθυνσης που δείχνει ο δείκτης στοίβας SP μεταφέρεται στον Α και κατόπιν ο SP μειώνεται κατά 1.

Κώδικας : 01001

Μνημονικό : POP

41. **Αποθήκευση του δείκτη στοίβας στη μνήμη (Store Stack Pointer)**

Αποθηκεύεται το περιεχόμενο του SP στη μνήμη.

Κώδικας : 11011

Μνημονικό : SSP

42. **Επιστροφή από υπορουτίνα (RETurn)**

    Επιστροφή από υπορουτίνα, δηλαδή ο PC παίρνει την τιμή της θέσης μνήμης όπου δείχνει ο SP και κατόπιν ο SP ελαττώνεται κατά 1.

    Κώδικας : 11110

    Μνημονικό : RET

3. Ανάλυση μικρολειτουργιών μονάδας ελέγχου

Η ακολουθία ελέγχου για τους τέσσερις κύκλους εντολής του TRN+ έχει ως εξής:

Κύκλος ανάκλησης εντολής

$f_0 t_0$: AR←PC;
$f_0 t_1$: Read; PC ←PC+1
$f_0 t_2$: IR ←BR; AR←BR(AP);
$f_0 t_3 \overline{ED}$: F1←1; F2←1;
$f_0 t_3 E\overline{D}$: F2←1;
$f_0 t_3 D$: F1←1;

Κύκλος δεικτοδοτημένης αναφοράς

$f_1 t_0 \overline{E}$: AR←IR(AP)+I; F2←1; CSC;
$f_1 t_0 E$: AR←IR(AP)+I; F2←1; F1←0; CSC;

Κύκλος έμμεσης αναφοράς

$f_2 t_0$: Read;
$f_2 t_1$: AR←BR(AP); F1←1; CSC;

Κύκλος εκτέλεσης εντολής

Παρακάτω παρουσιάζεται ένα πλήρες διάγραμμα των εντολών του TRN+, καθώς και η υλοποίησή τους σε επίπεδο μικρολειτουργίας:

Μνημονικό	Κώδικας Εντολής	Συνθήκη ελέγχου	Μικρολειτουργίες	Σημασία
NOP	00000	$i_0 f_3 t_0$	F1F2←00; SC←0;	Καμία Λειτουργία
LDA	00001	$i_1 f_3 t_0$	Read;	A←M[T]

		$i_1 f_3 t_1$	A←BR; F1F2←00; SC←0;	
LDX	00010	$i_2 f_3 t_0$ $i_2 f_3 t_1$	Read; X←BR; F1F2←00; SC←0;	X←M[T]
LDI	00011	$i_3 f_3 t_0$ $i_3 f_3 t_1$	Read; I←AP(BR); F1F2←00; SC←0;	I←M[T]
STA	00100	$i_4 f_3 t_0$ $i_4 f_3 t_1$	BR←A; Write; F1F2←00; SC←0;	M[T]←A
STX	00101	$i_5 f_3 t_0$ $i_5 f_3 t_1$	BR←X; Write; F1F2←00; SC←0;	M[T]←X
STI	00110	$i_6 f_3 t_0$ $i_6 f_3 t_1$	IC(BR)←0; AP(BR)←I; Write; F1F2←00; SC←0;	M[T]←I
ENA	00111	$i_7 f_3 t_0$	A←IR(AP); F1F2←00; SC←0;	A←IR(AP)
PSH	01000	$i_8 f_3 t_0$ $i_8 f_3 t_1$ $i_8 f_3 t_2$	SP←SP + 1; BR←A; AR←SP; Write; F1F2←00; SC←0;	M[SP++]←A
POP	01001	$i_9 f_3 t_0$ $i_9 f_3 t_1$ $i_9 f_3 t_2$	AR←SP; Read; A←BR; SP←SP–1 ; F1F2←00; SC←0;	A←M[SP--]
INA	01010	$i_{10} f_3 t_0 \overline{IR(2)}\ \overline{IR(1)}\ \overline{IR(0)}$	A←A + 1; F1F2←00; SC←0;	A←A + 1
INX	01010	$i_{10} f_3 t_0 \overline{IR(2)}\ \overline{IR(1)}\ IR(0)$	X←X + 1; F1F2←00; SC←0;	X←X + 1
INI	01010	$i_{10} f_3 t_0 \overline{IR(2)}\ IR(1)\ \overline{IR(0)}$	I←I + 1; F1F2←00; SC←0;	I←I + 1
DCA	01010	$i_{10} f_3 t_0 \overline{IR(2)}\ IR(1)\ IR(0)$	A←A – 1; F1F2←00; SC←0;	A←A - 1
DCX	01010	$i_{10} f_3 t_0\ IR(2)\ \overline{IR(1)}\ \overline{IR(0)}$	X←X – 1; F1F2←00; SC←0;	X←X - 1
DCI	01010	$i_{10} f_3 t_0\ IR(2)\ \overline{IR(1)}\ IR(0)$	I←I – 1; F1F2←00; SC←0;	I←I - 1
ENI	01011	$i_{11} f_3 t_0$	I←IR(AP); F1F2←0; SC←0;	I←IR(AP)
LSP	01100	$i_{12} f_3 t_0$ $i_{12} f_3 t_1$	Read; SP←BR(AP); F1F2←0; SC←0;	SP←M[T]
ADA	01101	$i_{13} f_3 t_0$ $i_{13} f_3 t_1$	Read; A←A + BR; F1F2←00; SC←0;	A←A + M[T]
SUB	01110	$i_{14} f_3 t_0$ $i_{14} f_3 t_1$ $i_{14} f_3 t_2$	Read; $BR \leftarrow \overline{BR}$; A←A + 1; A←A + BR; F1F2←00; SC←0;	A←A - M[T]
AND	01111	$i_{15} f_3 t_0$ $i_{15} f_3 t_1$	Read; A←A ^ BR; F1F2←00; SC←0;	A←A && M[T]
ORA	10000	$i_{16} f_3 t_0$ $i_{16} f_3 t_1$	Read; A←A V BR; F1F2←00; SC←0;	A←A || M[T]
XOR	10001	$i_{17} f_3 t_0$ $i_{17} f_3 t_1$	Read; A←A $\oplus$ BR; F1F2←00; SC←0;	A←A ^ M[T]
CMA	10010	$i_{18} f_3 t_0$	$A \leftarrow \overline{A}$; F1F2←00; SC←0;	$A \leftarrow \overline{A}$
JMP	10011	$i_{19} f_3 t_0$	PC←BR(AP); F1F2←00; SC←0;	PC←T
JPN	10100	$i_{20} f_3 t_0 S$ $i_{20} f_3 t_0 \overline{S}$	PC←BR(AP); F1F2←00; SC←0; F1F2←00; SC←0;	PC←T
JAG	10101	$i_{21} f_3 t_0 \overline{S}\ \overline{Z}$ $i_{21} f_3 t_0 (S+Z)$	PC← BR(AP); F1F2←00; SC←0; F1F2←00; SC←0;	PC←T
JPZ	10110	$i_{22} f_3 t_0 Z$ $i_{22} f_3 t_0 \overline{Z}$	PC← BR(AP); F1F2←0; SC←0; F1F2←00; SC←0;	PC←T
JPO	10111	$i_{23} f_3 t_0 V$	PC← BR(AP); F1F2←0; SC←0;	PC←T

		$i_{23} f_3 t_0 \overline{V}$	F1F2←00; SC←0;	
JSR	11000	$i_{24} f_3 t_0$ $i_{24} f_3 t_1$ $i_{24} f_3 t_2$	SP←SP + 1; AR←SP; BR(AP)←PC; Write; PC← IR(AP); F1F2←00; SC←0;	SP←PC(Jump to AP(IR))
JIG	11001	$i_{25} f_3 t_0$ (I>0) $i_{25} f_3 t_0$ (I≤0)	PC← BR(AP); F1F2←00; SC←0; F1F2←00; SC←0;	PC←T
SHAL	11010	$i_{26} f_3 t_0$ $\overline{IR(1)}$ $\overline{IR(0)}$	Shal; F1F2←00; SC←0;	A<<
SHAR	11010	$i_{26} f_3 t_0$ $\overline{IR(1)}$ IR(0)	Shar; F1F2←00; SC←0;	A>>
SHXL	11010	$i_{26} f_3 t_0$ IR(1) $\overline{IR(0)}$	Shxl; F1F2←00; SC←0;	X<<
SHXR	11010	$i_{26} f_3 t_0$ IR(1) IR(0)	Shxr; F1F2←00; SC←0;	X>>
SSP	11011	$i_{27} f_3 t_0$ $i_{27} f_3 t_1$	BR(AP)←SP; Write; F1F2←00; SC←0;	M[T]←SP
SAXL	11100	$i_{28} f_3 t_0$ $\overline{IR(0)}$	Shaxl; F1F2←00, SC←0;	AX<<
SAXR	11100	$i_{28} f_3 t_0$ IR(0)	Shaxr; F1F2←00, SC←0;	AX>>
INP	11101	$i_{29} f_3 t_0$ $\overline{IR(0)}$ $i_{29} f_3 t_1$ $\overline{IR(0)}$	Input; A←BR; F1F2←00; SC←0;	Θύρα Εισόδου←T
OUT	11101	$i_{29} f_3 t_0$ IR(0) $i_{29} f_3 t_1$ IR(0)	BR←A; Output; F1F2←00; SC←0;	Θύρα Εξόδου←A
RET	11110	$i_{30} f_3 t_0$ $i_{30} f_3 t_1$ $i_{30} f_3 t_2$	AR←SP; Read; PC← BR(AP); SP←SP–1; F1F2←00; SC←0;	Επιστροφή από υπορουτίνα
HLT	11111	$i_{31} f_3 t_0$	H←1	Παύση Λειτουργίας Επεξεργαστή

4. Προγραμματισμός του TRN+

Ο επεξεργαστής διαθέτει το σύνολο των 42 εντολών για να υλοποιηθεί οποιοδήποτε πρόγραμμα που φορτώνεται στη μνήμη. Ο χρήστης καλείται να γράψει το πρόγραμμα του στη συμβολική γλώσσα (assembly) του TRN+. Ακολουθεί ένα παράδειγμα :

```
NAM     Recursive
ORG     0
STR:    LDI   DATA
        LSP   DATA2
        JSR   SBU          // This is a Comment
STOP:   HLT
SBU:    DCI
        JSR   CPIA
        JPZ   STOP
        JSR   SBU
        RET
CPIA:   STI   TEMP
        LDA   TEMP
        RET
TEMP:   RES   1
DATA:   CON   2
DATA2:  CON   32
        RES   3
        END
```

Κανόνες σύνταξης προγραμμάτων

- Κάθε πρόγραμμα πρέπει να έχει ένα όνομα που δηλώνεται με την ψευδοεντολή NAM.
- Οι ψευδοεντολές που υποστηρίζει ο assembler είναι NAM,CON,RES,ORG,ENT και END. Η χρήση τους είναι πανομοιότυπη με αυτή του TRN.
- Οι εντολές και οι ψευδοεντολές γράφονται με κεφαλαία.
- Ο TRN υποστηρίζει έμμεση αναφορά [π.χ. LDA (TEMP)], δεικτοδοτημένη αναφορά [π.χ. LDA,I TEMP] καθώς και συνδυασμό και των δύο [π.χ. LDA,I (TEMP)].
- Κάθε πρόγραμμα πρέπει να έχει φυσικό τέλος HLT.
- Οι ετικέτες μπορούν να είναι οποιαδήποτε ακολουθία χαρακτήρων ΕΚΤΟΣ από την κωδική ονομασία των εντολών και των ψευδοεντολών . **ΠΡΕΠΕΙ** πάντα να ακολουθούνται από <**:**> κατά τον ορισμό τους.
- Το TAB και το SPACE είναι ισοδύναμα.
- Τα σχόλια μπαίνουν μετά τις εντολές ύστερα από <//> ή σε δική τους γραμμή.
- Παράβαση κάποιου από τα παραπάνω προκαλεί σφάλμα κατά τη μετάφραση του προγράμματος.

Αφού γραφεί το πρόγραμμα, το φορτώνουμε στη μνήμη του TRN (**File →Load Assembly File**) μέσα από τον προσομοιωτή. Αν προκύψει λάθος κατά τη μετάφραση του συμβολικού προγράμματος το πρόγραμμα δεν φορτώνεται στη μνήμη.

5. Χειρισμός του προσομοιωτή

Αφού φορτώσουμε το πρόγραμμα στη μνήμη έχουμε τη δυνατότητα

- Να εκτελέσουμε το πρόγραμμα (**Run**)
- Να διακόψουμε ή να παγώσουμε τη λειτουργία του (**Halt – Stop**)
- Να αποθηκεύσουμε την κατάσταση της μνήμης του επεξεργαστή
- Να ρυθμίσουμε την ταχύτητα εκτέλεσης- προσομοίωσης.
- Να καθορίσουμε το επίπεδο της ανάλυσης της προσομοίωσης ανά μικρολειτουργία ή ανά εντολή.

6. Τροποποιήσεις του TRN+ σε σχέση με τον TRN

Ο TRN+ είναι εφοδιασμένος με ένα καταχωρητή στοίβας (SP). Για αυτό παρέχονται στον προγραμματιστή 6 νέες εντολές (LSP, SSP, PUS, POP, JSR και RET) που αποσκοπούν στον χειρισμό της στοίβας. Πριν χρησιμοποιήσουμε τον καταχωρητή στοίβας θα πρέπει να τον αρχικοποιήσουμε ώστε να δείχνει σε κάποια περιοχή της μνήμης (βλέπε παράδειγμα πιο πάνω: LSP DATA2). Οι εντολές JSR και RET χρησιμοποιούν τη στοίβα για την αποθήκευση της διεύθυνσης επιστροφής, επιτρέποντας έτσι την εκτέλεση και αναδρομικών ρουτινών. Κάθε JSR θα πρέπει να αντιστοιχεί και σε μια εντολή RET. Η εντολή SSP αποθηκεύει το περιεχόμενο του δείκτη στοίβας στη μνήμη. Η PSH και η POP προσθέτουν και αφαιρούν στοιχεία από τη στοίβα, αντίστοιχα. Ακόμα έχουν δημιουργηθεί οι εντολές SHAL, SHAR, SHXL και SHXR για την ολίσθηση των καταχωρητών A και X . Επίσης για τον καταχωρητή AX υπάρχουν οι αντίστοιχες SAXL και SAXR. Τέλος υπάρχουν εντολές για την αύξηση (INA, INX, INI) και μείωση (DCA, DCX, DCI) των καταχωρητών A , X και I.

7. Εγκατάσταση του Λογισμικού

Για να καταστεί δυνατή η εκτέλεση του λογισμικού προσομοίωσης είναι απαραίτητη η εγκατάσταση του Java Virtual Machine της Sun. Στη συνέχεια, αφού κατεβάσουμε το αρχείο trn.zip από το site του μαθήματος πραγματοποιούμε τα ακόλουθα βήματα προκειμένου να εγκατασταθεί το λογισμικό:

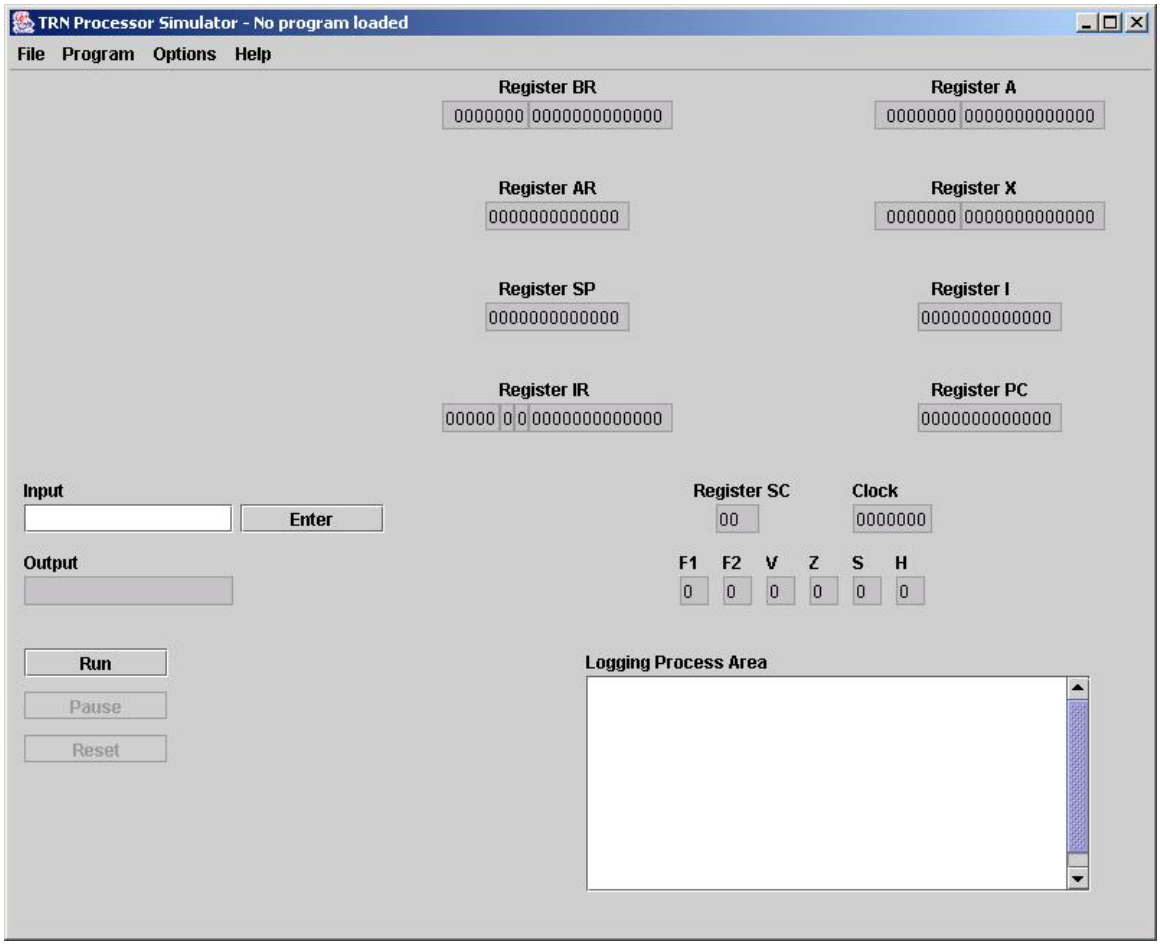
1. Αποσυμπιέζουμε το αρχείο trn.zip στο σκληρό δίσκο C:. Έτσι, δημιουργείται το ακόλουθο path: C:\trn\
στο οποίο περιέχονται το αρχείο trn.jar και το αρχείο trn.bat.
2. Στη συνέχεια κάνουμε διπλό κλικ στο αρχείο trn.bat και η εφαρμογή εκτελείται αν έχουμε εγκαταστήσει σωστά τη java. Σε περίπτωση όπου έχουμε εγκαταστήσει το JRE (Java Runtime Enviroment) με διπλό click πάνω στο trn.jar η εφαρμογή θα ξεκινήσει. Το JRE διατίθεται και αυτό από το site του μαθήματος.

.

8. Παράδειγμα χρήσης του προσομοιωτή του TRN+

Παρακάτω αναφέρεται ένα ολοκληρωμένο σενάριο χρήσης του λογισμικού προσομοίωσης του TRN+.

1. Αρχικά, ανοίγουμε το περιβάλλον προσομοίωσης που απεικονίζεται στην εικόνα 1. Το περιβάλλον ανοίγει κάνοντας διπλό κλικ στο αρχείο **trn.bat**. Όπως, παρατηρούμε στη γραμμή τίτλου του παραθύρου, αρχικά δε φορτώνεται κανένα πρόγραμμα στον προσομοιωτή (“No program loaded.”). Διακρίνουμε, επίσης, όλους τους καταχωρητές της κεντρικής μονάδας επεξεργασίας, το πεδίο εισαγωγής δεδομένων (“Input”) και την περιοχή καταγραφής γεγονότων (“Logging Process Area”).

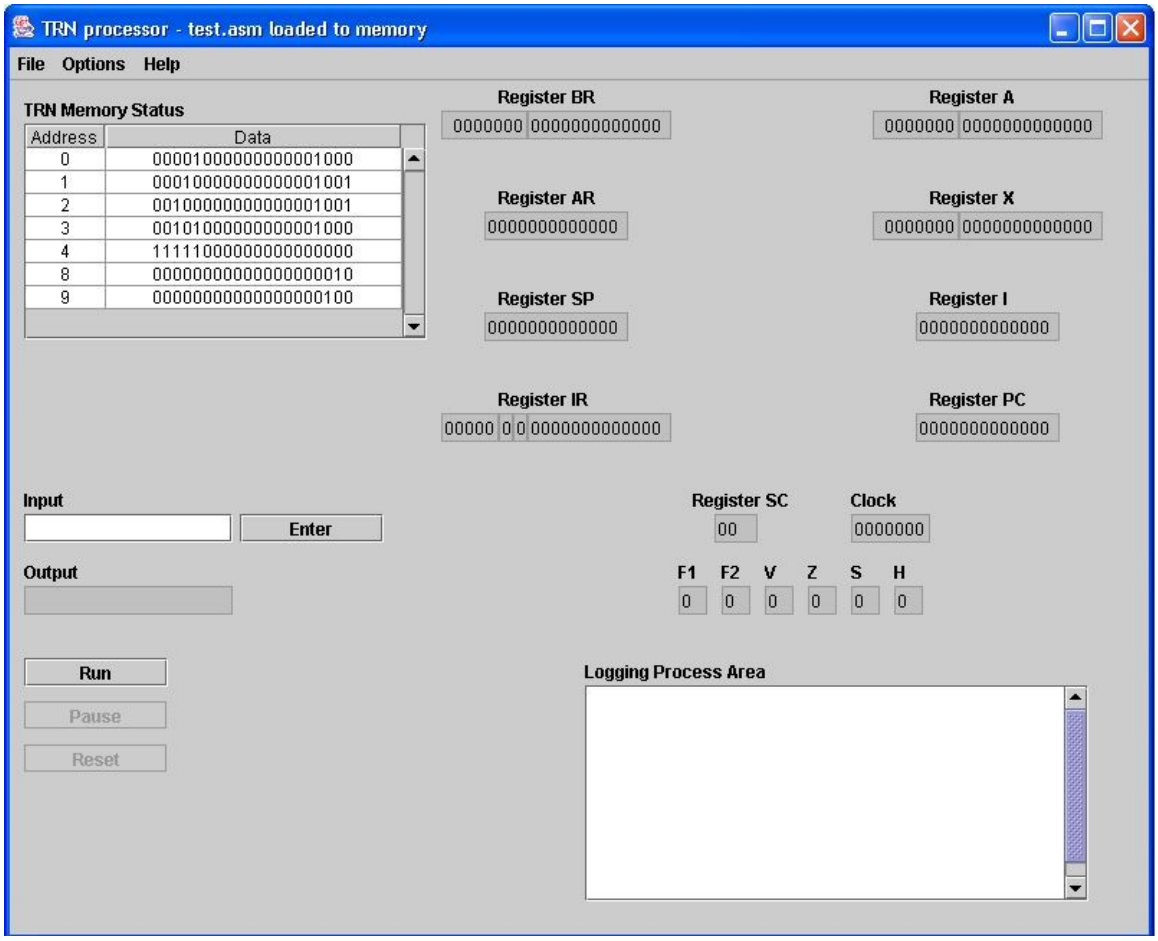


Εικόνα 1. Περιβάλλον προσομοιωτή

2. Ανοίγουμε το Notepad ή οποιοδήποτε αλλό επεξεργαστή κειμένου προκειμένου να γράψουμε την εφαρμογή μας σε γλώσσα Assembly. Σώζουμε το αρχείο με κατάληξη ".asm". Στη συνέχεια, εκτελούμε την παρακάτω ακολουθία ενεργειών μέσα από το περιβάλλον προσομοιώσης.

   File → Open Assembly File
   Εάν το πρόγραμμά μας είναι ορθό συντακτικώς τότε φορτώνεται στη μνήμη και είναι έτοιμο προς εκτέλεση – προσομοίωση .
   Σε περίπτωση εύρεσης συντακτικού λάθους, λαμβάνουμε μήνυμα λάθους συμβολομετάφρασης (Translation Error) και το πρόγραμμα δεν φορτώνεται στη μνήμη.

3. Εφόσον το πρόγραμμα δεν παρουσιάσει συντακτικό λάθος, φορτώνεται στη μνήμη και απεικονίζεται στην πάνω αριστερή γωνία του περιβάλλοντος προσομοίωσης με τη μορφή «Διεύθυνση - Δεδομένα»



Εικόνα 2. Κατάσταση μνήμης του προσομοιωτή

4. Για την εκκίνηση προσομοίωσης πατάμε “Run”.
5. Καθώς το πρόγραμμα εκτελείται, έχουμε τη δυνατότητα
    i. να τροποποιήσουμε την ταχύτητα προσομοίωσης (slow, medium, fast, define).
    ii. να παγώσουμε την εκτέλεση του προγράμματος (Pause)
    iii. να διακόψουμε την εκτέλεση του προγράμματος (Stop)
6. Σε περίπτωση που θέλουμε να αποθηκεύσουμε την κατάσταση της μνήμης του επεξεργαστή παγώνουμε το πρόγραμμα (Pause) και στη συνέχεια εκτελούμε την παρακάτω ακολουθία ενεργειών

File → Save memory to image file

Αντίστροφα, έχουμε τη δυνατότητα να επανακτήσουμε το στιγμιότυπο της μνήμης οποτεδήποτε εμείς επιλέξουμε.

File → Load memory image file (“.mif”)

7. Αν παρουσιαστεί λάθος κατά την εκτέλεση του προγράμματος (“Runtime Error”) ο χρήστης ειδοποιείται με μήνυμα λάθους.

8. Μετά το πέρας της προσομοίωσης μπορούμε να τυπώσουμε το περιεχόμενο της περιοχής καταγραφής γεγονότων.