**Ασκήσεις πάνω στη σχεδίαση εντολών επεξεργαστή**

1. Να γράψετε την ανάκληση των εντολών και να σκεφτείτε τι συμβαίνει στο βήμα T1, δηλαδή ποια είναι η πληροφορία την οποία λαμβάνει ο καταχωρητής MDR

2. Για κάθε άσκηση, είναι χρήσιμο να κάνετε ένα διάγραμμα με τη ροή των πληροφοριών που μεταφέρονται ανάμεσα στους καταχωρητές. Θα σας βοηθήσει να επιλύσετε τις ασκήσεις που ακολουθούν.
3. Ένας υπολογιστής έχει εντολές της παρακάτω μορφής:

| OPCODE | ADDRESS |
|---|---|

Έστω ότι θέλουμε να σχεδιάσουμε την εντολή SWAP A, ενός παράγοντα. Η εντολή αυτή εναλλάσσει τα περιεχόμενα της θέσης μνήμης A, με τα περιεχόμενα της κορυφής της στοίβας, στην οποία δείχνει ο SP. Χρησιμοποιήστε RTL για να δείξετε την εκτέλεση της εντολής με αρχιτεκτονική ενός διαύλου. Το μέγεθος της στοίβας δεν μεταβάλλεται.

**ΛΥΣΗ**

| **Ανάκληση SWAP A** | **Επεξήγηση** |
|---|---|
| $T_0$: MAR ← PC, Z ← PC+1 | $T_0$: Ο PC περιέχει την διεύθυνση της εντολής που θα εκτελεστεί. Την δίνει στην MAR. Ο Z αποθηκεύει την τιμή του PC+1 |
| $T_1$: MDR ← M[MAR], PC ← Z | $T_1$: Αποθηκεύεται στον MDR η εντολή SWAP (PCODE\|ADDRESS). Ο PC αυξάνεται κατά 1 ώστε να δείχνει στην επόμενη εντολή. |
| $T_2$: IR ← MDR | $T_2$: Έστω IR και MDR ίσοι στο μέγεθος, οπότε αποθηκεύεται όλος ο MDR στον IR. |
| **ΕΚΤΕΛΕΣΗ SWAP A** | |
| $T_3$: MAR ← IR(ADDRESS) | $T_3$: MAR αποθηκεύει την δ. του A |
| $T_4$: MDR ← M[MAR] | $T_4$: MDR αποθηκεύει τον A (τα data που βρίσκονται στη δ του A) |
| $T_5$: ACC ← MDR | $T_5$: Ο A αποθηκεύεται προσωρινά στον ACC γιατί θα χρειαστούμε ξανά τον MDR |
| $T_6$: MDR ← M[MAR], Z ← SP+1 | $T_6$: MAR αποθηκεύει τη δ. Που δείχνει ο SP |
| $T_7$: MDR ← M[MAR], SP ← Z | $T_7$: MDR αποθηκεύει τα data που είναι αποθηκευμένα στην δ. Που δείχνει ο SP |
| $T_8$: MAR ← IR(ADDRESS) | $T_8$: Ο MAR ξαναπαίρνει την δ. του A από τον IR, αυτή τη φορά για να γράψουμε |

| | |
|---|---|
| | στην δ. αυτή |
| $T_9$: M[MAR] ← MDR | $T_9$: MDR γράφει στη μνήμη, στη δ του Α, τα data που αποθήκευσε προηγουμένως από τη στοίβα |
| $T_{10}$: Z ← SP-1 | $T_{10}$: SP μειώνεται για να δείχνει στην επόμενη ελεύθερη θέση |
| $T_{11}$: SP ← Z, MAR ← Z | $T_{11}$: SP αποθηκεύεται στον MAR |
| $T_{12}$: MDR ← ACC | $T_{12}$: MDR λαμβάνει τον Α από τον ACC |
| $T_{13}$: M[MAR] ← MDR | $T_{13}$: MDR γράφει στη στοίβα |

4. Ένας υπολογιστής του ενός εσωτερικού διαύλου (bus) έχει εντολές της παρακάτω μορφής:

| OPCODE | ADDRESS |
|---|---|

Δώστε την εκτέλεση της εντολής POPA. Η εντολή αυτή εξάγει την τιμή που υπάρχει στην κορυφή της στοίβας (stack) και την τοποθετεί στην θέση μνήμης Α. Η κορυφή της στοίβας περιέχεται στον καταχωρητή SP ο οποίος θα πρέπει να αυξηθεί κατά 1 αφού εξαχθεί ο παράγοντας. Χρησιμοποιήστε RTL για να δείξετε την εκτέλεση της εντολής με αρχιτεκτονική ενός διαύλου.

**ΛΥΣΗ**

| Ανάκληση POP A | Επεξήγηση |
|---|---|
| $T_0$: MAR ← PC, Z ← PC+1 | |
| $T_1$: MDR ← M[MAR] | |
| $T_2$: IR ← MDR | |
| Εκτέλεση POP A | |
| $T_3$: MAR ← SP, Z ← SP+1 | $T_3$: Ξεκινά η ανάγνωση από την στοίβα δίνοντας την δ που δείχνει ο SP στον MAR |
| $T_4$: MDR ← M[MAR] | $T_4$: Ο MDR λαβάνει τα data που βρισκόντουσαν στην κορυφή της στοίβας και ο SP δείχνει πλέον στην επόμενη θέση της στοίβας. |
| $T_5$: MAR ← IR(ADDRESS) | $T_5$: Ο MAR αποθηκεύει τη δ του Α |
| $T_6$: M[MAR] ← MDR | $T_6$: Ο MDR γράφει στην μνήμη, στην δ του Α τα data που αποθήκευσε από την στοίβα |

5. Ένας υπολογιστής έχει εντολές της παρακάτω μορφής:

| OPCODE | ADDRESS |
|---|---|

Έστω ότι θέλουμε να σχεδιάσουμε την εντολή PUSH A. Η εντολή αυτή τοποθετεί στην κορυφή της στοίβας τα δεδομένα που υπάρχουν στη θέση μνήμης A. Η θέση υποδοχής της στοίβας (κορυφή της στοίβας) υπάρχει στον καταχωρητή SP, ο οποίος μειώνεται κατά 1 αφού τοποθετηθεί ο παράγοντας στη στοίβα. Χρησιμοποιείστε RTL για να δείξετε την εκτέλεση της εντολής με αρχιτεκτονική ενός διαύλου.

**ΛΥΣΗ**

| Ανάκληση PUSH A | Επεξήγηση |
|---|---|
| $T_0$: MAR ← PC, Z ← PC+1 | |
| $T_1$: MDR ← M[MAR], PC ← Z | |
| $T_2$: IR ← MDR | |
| Εκτέλεση PUSH A | |
| $T_3$: MAR ← IR(ADDRESS) | $T_3$: Ο MAR αποθηκεύει την δ. του A |
| $T_4$: MDR ← M[MAR] | $T_4$: Ο MDR αποθηκεύει τα data του A |
| $T_5$: Z ← SP-1 | $T_5$: Ο SP μειώνεται ώστε να δείχνει στην επόμενη ελεύθερη θέση |
| $T_6$: SP ← Z, MAR ← Z | $T_6$: Ο MDR γράφει στην στοίβα τον A |
| $T_7$: M[MAR] ← MDR | |

6. Ένας υπολογιστής του ενός εσωτερικού διαύλου (bus) έχει εντολές της παρακάτω μορφής:

| OPCODE | ADDRESS1 | ADDRESS2 |
|---|---|---|

Δώστε την εκτέλεση της εντολής SWAPAB. Η εντολή αυτή εναλλάσσει τα περιεχόμενα των θέσεων μνήμης A και B που δίνονται στα πεδία ADDRESS1 και ADDRESS2 αντίστοιχα. Χρησιμοποιείστε RTL για να δείξετε την εκτέλεση της εντολής.

7. Ένας υπολογιστής έχει εντολές της παρακάτω μορφής:

| OPCODE | ADDRESS1 | ADDRESS2 |
|---|---|---|

Δώστε την εκτέλεση της εντολής SWAP_STACK_AB. Η εντολή αυτή εναλλάσσει τα περιεχόμενα των θέσεων μνήμης A και B που δίνονται στα πεδία ADDRESS1 και ADDRESS2, χωρίς να χρησιμοποιεί κάποιο καταχωρητή για ενδιάμεση αποθήκευση, εκτός φυσικά από τους απαιτούμενους για επικοινωνία με τη μνήμη. Ειδικότερα, τα περιεχόμενα της διεύθυνσης A μεταφέρονται στην πρώτη ελεύθερη θέση της στοίβας, στη συνέχεια τα περιεχόμενα της B μεταφέρονται στην A και τέλος τα περιεχόμενα της στοίβας μεταφέρονται στη B. Χρησιμοποιείστε RTL για να δείξετε την εκτέλεση της εντολής θεωρώντας αρχιτεκτονική ενός εσωτερικού διαύλου.

8. Να σχεδιάσετε την εντολή CMP A, η οποία συγκρίνει την τιμή που βρίσκεται στη θέση μνήμης A με μία σταθερά, η οποία αποθηκεύεται στη στοίβα, στη θέση που δείχνει ο SP. Χρησιμοποιήστε RTL για να δείξετε την εκτέλεση της εντολής με αρχιτεκτονική ενός διαύλου.

9. Να σχεδιάσετε την εντολή CMP, η οποία διαβάζει διαδοχικά τις δύο τιμές που υπάρχουν στην κορυφή της στοίβας και τις συγκρίνει μεταξύ τους. Χρησιμοποιήστε RTL για να δείξετε την εκτέλεση της εντολής με αρχιτεκτονική ενός διαύλου.

10. Να σχεδιάσετε την εντολή MUL A, η οποία πολλαπλασιάζει τα περιεχόμενα της θέσης μνήμης A με τα περιεχόμενα του ACC και αποθηκεύει το αποτέλεσμα στη θέση μνήμης A. Χρησιμοποιήστε RTL για να δείξετε την εκτέλεση της εντολής με αρχιτεκτονική ενός διαύλου.

11. Επαναλάβετε τα ερωτήματα 1, 3-10 για αρχιτεκτονική 2 εσωτερικών διαύλων.

12. Δίνεται ένας υπολογιστής του οποίου οι εντολές έχουν τη μορφή

| OPCODE | ADDRESS1 | ADDRESS2 | ADDRESS3 |
|---|---|---|---|

όπου κάθε πεδίο είναι 4 bits.
Έστω ότι επιθυμούμε να έχουμε 15 εντολές τριών παραγόντων, 15 εντολές δύο παραγόντων, 7 εντολές ενός παράγοντα και 144 εντολές με 0 παράγοντες, χρησιμοποιώντας την τεχνική Expanded Opcode. Είναι εφικτό; Αν ναι, να δείξετε σε δυαδική μορφή το πρώτο και το τελευταίο opcode κάθε μίας από τις τέσσερις παραπάνω μορφές. Να σχεδιάσετε ένα κύκλωμα, το οποίο επιστρέφει τιμή 1, αν μία εντολή είναι ενός παράγοντα.