

5.4. ΤΥΠΟΙ ΕΝΤΟΛΩΝ

Οι εντολές του επιπέδου συμβατικής μηχανής μπορεί κατά προσέγγιση να χωριστούν σε δύο ομάδες: εντολές γενικής χρήσης και εντολές ειδικής χρήσης. Για παράδειγμα, η ικανότητα μετακίνησης δεδομένων σε μια μηχανή απαιτείται σχεδόν από όλες τις εφαρμογές. Οι εντολές ειδικής χρήσης έχουν πολύ λιγότερες εφαρμογές. Για παράδειγμα, η εντολή `MOVEP` του 68000 αποθηκεύει το περιεχόμενο ενός καταχωρητή `D` σε αλληλοδιαδοχικά `byte` της μνήμης, με ένα αχρησιμοποίητο `byte` μεταξύ των `byte` δεδομένων. Λίγες εφαρμογές μπορεί να χρησιμοποιήσουν αποδοτικά αυτή την εντολή και κανένας μεταγλωττιστής δεν πρόκειται ποτέ να την παραγάγει. (Συμπεριλήφθηκε με σκοπό την απλοποίηση της επικοινωνίας με παλιότερα περιφερειακά τσιπ εύρους 8 bit.) Στις επόμενες ενότητες θα εξετάσουμε τις κύριες ομάδες των εντολών γενικής χρήσης.

5.4.1. Εντολές Μεταφοράς Δεδομένων

Η αντιγραφή δεδομένων από μια θέση σε άλλη είναι η πιο βασική από όλες τις λειτουργίες. Αντιγραφή σημαίνει δημιουργία ενός νέου αντικειμένου, με το ίδιο ακριβώς μοτίβο bit με το πρωτότυπο. Αυτή η χρήση της λέξης "μεταφορά" είναι κάπως διαφορετική από τη συνηθισμένη χρήση στην καθομιλουμένη γλώσσα. Όταν λέμε ότι ένα κατάστημα μεταφέρθη-

κε από την Ερμού στη Σταδίου δεν εννοούμε ότι ένα ακριβές αντίγραφο του καταστήματος μεταφέρθηκε στη Σταδίου και το πρωτότυπο παρέμεινε στην Ερμού. Όταν λέμε ότι το περιεχόμενο της θέσης μνήμης 2000 μεταφέρθηκε σε κάποιον καταχωρητή, σχεδόν πάντα εννοούμε ότι ένα ακριβές αντίγραφο δημιουργήθηκε εκεί και ότι το πρωτότυπο παρέμεινε όπως είναι στη θέση 2000. Οι εντολές μεταφοράς δεδομένων θα ονομάζονταν καλύτερα εντολές "αντιγραφής δεδομένων", αλλά ο όρος "μεταφορά δεδομένων" έχει ήδη καθιερωθεί.

Τα δεδομένα μπορούν να αποθηκευτούν σε πολλές θέσεις, που διαφέρουν ως προς τον τρόπο προσπέλασης των λέξεων. Τρεις συνηθισμένες θέσεις είναι σε μια συγκεκριμένη λέξη μνήμης, σ' έναν καταχωρητή, ή στη στοίβα. Η στοίβα βρίσκεται είτε σε ειδικούς καταχωρητές είτε στη μνήμη, αλλά ο τρόπος προσπέλασής της διαφέρει από τη συνηθισμένη προσπέλαση της μνήμης. Μια προσπέλαση στη μνήμη χρειάζεται μια διεύθυνση, ενώ κατά την τοποθέτηση ενός στοιχείου στη στοίβα δεν προσπελάζεται άμεσα η στοίβα. Στις εντολές μεταφοράς δεδομένων, η προέλευση των πληροφοριών (δηλαδή, η θέση των πρωτότυπων πληροφοριών) και ο προορισμός των πληροφοριών (δηλαδή, η θέση όπου θα τοποθετηθεί το αντίγραφο) πρέπει να καθορίζονται είτε άμεσα είτε έμμεσα.

Οι εντολές μεταφοράς δεδομένων πρέπει να υποδηλώνουν με κάποιον τρόπο την ποσότητα των δεδομένων που θα μεταφερθούν. Υπάρχουν εντολές για τη μεταφορά ποσοτήτων δεδομένων από 1 bit μέχρι ολόκληρη τη μνήμη. Στις μηχανές λέξεων σταθερού μήκους, το πλήθος των λέξεων που μεταφέρονται συνήθως καθορίζεται από την εντολή — για παράδειγμα, ξεχωριστές εντολές για τη μεταφορά μιας λέξης και τη μεταφορά μισής λέξης. Οι μηχανές λέξεων μεταβλητού μήκους συνήθως διαθέτουν εντολές που καθορίζουν μόνον τις διευθύνσεις προέλευσης και προορισμού αλλά όχι την ποσότητα. Η μεταφορά συνεχίζεται μέχρι να εντοπιστεί ένα σημάδι τέλους δεδομένων στα ίδια τα δεδομένα.

Ο 680x0 έχει μια εντολή γενικής χρήσης MOVE, με δύο αυθαίρετους τελεστές, όπως φαίνεται στην Εικόνα 5-23. Μπορεί να μεταφέρει δεδομένα μεταξύ καταχωρητών, μνήμης, ή της στοίβας οπουδήποτε. Οι CPU της Intel έχουν πολύ πιο περιορισμένες εντολές MOVE, αλλά υπάρχουν πολλές τέτοιες εντολές, οπότε είναι κι εδώ δυνατή η μεταφορά δεδομένων οπουδήποτε.

5.4.2. Δυαδικές Πράξεις

Δυαδικές πράξεις είναι εκείνες που συνδυάζουν δύο τελεστές για να δώσουν ένα αποτέλεσμα. Σχεδόν όλες οι μηχανές επιπέδου 2 έχουν εντολές για την εκτέλεση πρόσθεσης και αφαίρεσης μεταξύ ακεραίων. Εκτός από τους μικροϋπολογιστές των 8 bit, ο πολλαπλασιασμός και η διαίρεση ακεραίων είναι, επίσης, βασικές πράξεις. Υποθέτουμε ότι είναι περιττό να εξηγήσουμε γιατί οι υπολογιστές διαθέτουν εντολές αριθμητικών πράξεων.

Μια άλλη ομάδα δυαδικών πράξεων είναι οι λογικές εντολές (Boolean). Αν και υπάρχουν 16 λογικές συναρτήσεις δύο μεταβλητών, λίγες μηχανές επιπέδου 2 έχουν εντολές και για τις 16. Για παράδειγμα, η συνάρτηση που υπολογίζει την τιμή ΑΛΗΘΕΣ (TRUE), ανεξάρτητα από ορίσματα, είναι άχρηστη. Αν το μοτίβο bit της τιμής ΑΛΗΘΕΣ χρειάζεται σε κάποιο σημείο της μηχανής, μπορεί απλώς να μεταφερθεί εκεί αντί να υπολογιστεί. Παρόμοια

επιχειρήματα εξηγούν γιατί μια εντολή για την υλοποίηση της συνάρτησης $f(P, Q) = P$ είναι μάλλον άχρηστη.

Τρεις εντολές που υπάρχουν σε πολλές μηχανές είναι οι AND (ΚΑΙ), OR (Η), και EXCLUSIVE OR (ΑΠΟΚΛΕΙΣΤΙΚΟ Η). Στις μηχανές λέξεων σταθερού μήκους, η εντολή AND υπολογίζει bit προς bit το λογικό γινόμενο δύο ορισμάτων μίας λέξης, το αποτέλεσμα του οποίου είναι επίσης μια λέξη. Συχνά υπάρχουν, επίσης, εντολές για byte και διπλές λέξεις. Παρόμοιες παρατηρήσεις ισχύουν και για τις άλλες λογικές πράξεις.

Μια σημαντική χρήση της AND είναι η εξαγωγή bit από λέξεις. Ας θεωρήσουμε, για παράδειγμα, μια μηχανή λέξεων μήκους 32 bit όπου αποθηκεύονται τέσσερις χαρακτήρες των 8 bit ανά λέξη. Ας υποθέσουμε ότι χρειάζεται να ξεχωρίσουμε το δεύτερο χαρακτήρα από τους άλλους τρεις για να τον τυπώσουμε — δηλαδή, πρέπει να δημιουργήσουμε μία λέξη που να περιέχει αυτόν το χαρακτήρα στα 8 δεξιότερα bit (ευθυγραμμισμένο δεξιά — right justified), και μηδέν στα 24 αριστερότερα bit.

Για να εξαγάγουμε αυτόν το χαρακτήρα, εκτελούμε την πράξη AND σ' αυτόν το χαρακτήρα και σε μια σταθερά, που λέγεται μάσκα (mask). Σαν αποτέλεσμα αυτής της πράξης, τα bit που δε χρειαζόμαστε αλλάζουν σε μηδέν, όπως φαίνεται παρακάτω.

```
10110111 10111100 11011011 10001011 A
00000000 11111111 00000000 00000000 B (μάσκα)
00000000 10111100 00000000 00000000 A AND B
```

Το αποτέλεσμα στη συνέχεια ολισθαίνει 16 bit προς τα δεξιά ώστε να απομονωθεί ο χαρακτήρας στο δεξιό άκρο της λέξης.

Μια σημαντική χρήση της OR είναι η εισαγωγή συγκεκριμένων bit σε μια λέξη, λειτουργία αντίθετη της εξαγωγής. Για να αλλάξουμε τα 8 δεξιότερα bit μιας λέξης των 32 bit χωρίς να πειράξουμε τα υπόλοιπα 24 bit, πρώτα μηδενίζονται με χρήση μάσκας τα 8 bit που θέλουμε να αλλάξουμε, και στη συνέχεια εισάγεται ο νέος χαρακτήρας με την πράξη OR, όπως φαίνεται παρακάτω.

```
10110111 10111100 11011011 10001011 A
11111111 11111111 11111111 00000000 B (μάσκα)
10110111 10111100 11011011 00000000 A AND B
00000000 00000000 00000000 01010111 C
10110111 10111100 11011011 01010111 (A AND B) OR C
```

Η πράξη AND τείνει να αφαιρεί bit 1, επειδή ποτέ δεν υπάρχουν περισσότερα bit 1 στο αποτέλεσμα απ' ό,τι σε οποιονδήποτε από τους δύο τελεστές. Η πράξη OR τείνει να εισάγει bit 1, επειδή υπάρχουν πάντα τουλάχιστον τόσα bit 1 στο αποτέλεσμα όσα στον τελεστέο με τα περισσότερα bit 1. Η πράξη EXCLUSIVE OR, από την άλλη μεριά, είναι συμμετρική, τείνοντας, κατά το μέσο όρο, να μην εισάγει ούτε να αφαιρεί bit 1. Αυτή η συμμετρία όσον αφορά στα bit 1 και 0 είναι χρήσιμη σε μερικές περιπτώσεις — για παράδειγμα, κατά την παραγωγή "τυχαίων αριθμών".

Οι πρώτοι υπολογιστές εκτελούσαν πράξεις κινητής υποδιαστολής καλώντας διαδικασίες λογισμικού, αλλά σήμερα πολλοί υπολογιστές, ειδικότερα αυτοί που προορίζονται για επιστημονικές εργασίες, διαθέτουν εντολές κινητής υποδιαστολής στο επίπεδο 2 για λόγους ταχύτητας. Μερικές μηχανές παρέχουν πολλά μήκη αριθμών κινητής υποδιαστολής, τα μικρότερα για αυξημένη ταχύτητα και τα μεγαλύτερα για περιπτώσεις όπου χρειάζεται ακρίβεια πολλών ψηφίων. Οι αριθμοί κινητής υποδιαστολής εξετάζονται στο Παράρτημα Β.

5.4.3. Μοναδιαίες Πράξεις

Οι μοναδιαίες πράξεις χρησιμοποιούν έναν τελεστέο και παράγουν ένα αποτέλεσμα. Επειδή πρέπει να καθοριστεί μία διεύθυνση λιγότερο απ' ό,τι στις δυαδικές πράξεις, οι εντολές είναι μερικές φορές πιο σύντομες.

Οι εντολές ολίσθησης ή περιστροφής του περιεχομένου μιας λέξης ή byte είναι πολύ χρήσιμες και συχνά παρέχονται σε αρκετές διαφορετικές παραλλαγές. Οι ολισθήσεις είναι πράξεις που μετακινούν τα bit προς τα αριστερά ή προς τα δεξιά: τα bit που ολισθαίνουν πέρα από το άκρο της λέξης χάνονται. Οι περιστροφές είναι ολισθήσεις στις οποίες τα bit που εκτοπίζονται από το ένα άκρο επανεμφανίζονται στο άλλο. Η διαφορά μεταξύ μιας ολίσθησης και μιας περιστροφής απεικονίζεται παρακάτω.

```
00000000 00000000 00000000 01110011 A
```

```
00000000 00000000 00000000 00011100 Ολίσθηση του A δύο bit δεξιά
```

```
11000000 00000000 00000000 00011100 Περιστροφή του A δύο bit δεξιά
```

Και οι αριστερές και οι δεξιές ολισθήσεις και περιστροφές είναι χρήσιμες. Αν μια λέξη των n bit περιστραφεί προς τα αριστερά κατά k bit, το αποτέλεσμα είναι το ίδιο σαν να είχε περιστραφεί προς τα δεξιά κατά $n-k$ bit.

Οι δεξιές ολισθήσεις συχνά εκτελούνται με επέκταση προσήμου. Αυτό σημαίνει ότι οι θέσεις που εκκενώνονται στα αριστερά άκρο της λέξης γεμίζουν με το αρχικό bit προσήμου, 0 ή 1. Είναι σαν το bit προσήμου να σύρεται προς τα δεξιά. Μεταξύ άλλων, σημαίνει ότι ένας αρνητικός αριθμός θα παραμείνει αρνητικός. Αυτή η περίπτωση απεικονίζεται παρακάτω για ολισθήσεις προς τα δεξιά κατά 2 bit.

```
11111111 11111111 11111111 11110000 A
```

```
00111111 11111111 11111111 11111100 Ολίσθηση του A χωρίς επέκταση προσήμου
```

```
11111111 11111111 11111111 11111100 Ολίσθηση του A με επέκταση προσήμου
```

Μια σημαντική χρήση της ολίσθησης είναι ο πολλαπλασιασμός και η διαίρεση με δυνάμεις του 2. Αν ένας θετικός ακέραιος ολισθήσει αριστερά κατά k bit, το αποτέλεσμα, χωρίς να καλύπτομε την περίπτωση υπερχείλισης, είναι ο αρχικός αριθμός πολλαπλασιασμένος με το 2^k . Αν ένας θετικός ακέραιος ολισθήσει κατά k bit, το αποτέλεσμα είναι ο αρχικός αριθμός διαιρεμένος με το 2^k .

Η ολίσθηση μπορεί να χρησιμοποιηθεί για την επιτάχυνση ορισμένων αριθμητικών πράξεων. Ας θεωρήσουμε, για παράδειγμα, τον υπολογισμό $18 \times n$ για κάποιο θετικό ακέραιο n . Επειδή $18 \times n = 16 \times n + 2 \times n$, το $16 \times n$ μπορεί να επιτευχθεί με ολίσθηση προς τα

αριστερά κατά 4 bit ενός αντιγράφου του αριθμού n . Το $2 \times n$ μπορεί να επιτευχθεί με ολίσθηση προς τα αριστερά κατά 1 bit του αριθμού n . Το άθροισμα αυτών των δύο αριθμών ισούται με $18 \times n$. Οι πολλαπλασιασμός επιτεύχθηκε με μια μεταφορά, δύο ολισθήσεις, και μια πρόσθεση, που συχνά εκτελούνται πιο γρήγορα από έναν πολλαπλασιασμό.

Η ολίσθηση αρνητικών αριθμών, ακόμη και με επέκταση προσήμου, δίνει εντελώς διαφορετικά αποτελέσματα, ωστόσο. Ας θεωρήσουμε, για παράδειγμα, τον αριθμό συμπληρώματος ως προς ένα, -1. Ολισθαίνοντας προς τα αριστερά κατά 1 bit μας δίνει τον αριθμό -3. Ολισθαίνοντας προς τα αριστερά κατά ένα bit ακόμα μας δίνει το -7:

```
11111111 11111111 11111111 11111110 -1 σε συμπλήρωμα ως προς 1
```

```
11111111 11111111 11111111 11111100 Ολίσθηση του -1 αριστερά κατά 1 bit = -3
```

```
11111111 11111111 11111111 11111000 Ολίσθηση του -1 αριστερά κατά 2 bit = -7
```

Η ολίσθηση αρνητικών αριθμών συμπληρώματος ως προς 1 προς τα αριστερά δεν τους πολλαπλασιάζει επί 2. Ωστόσο, η ολίσθηση προς τα δεξιά προσομοιώνει σωστά τη διαίρεση.

Ας θεωρήσουμε τώρα μια αναπαράσταση ως προς 2 του -1. Όταν ολισθαίνει προς τα δεξιά κατά 6 bit με επέκταση προσήμου, δίνει τον αριθμό -1, που είναι λάθος γιατί το ακέραιο μέρος του $-1/64$ είναι 0:

```
11111111 11111111 11111111 11111111 -1 σε συμπλήρωμα ως προς 2
```

```
11111111 11111111 11111111 11111111 Ολίσθηση του -1 προς τα δεξιά κατά 6 bit = -1
```

Η ολίσθηση προς τα αριστερά προσομοιώνει, ωστόσο, τον πολλαπλασιασμό επί 2.

Οι πράξεις περιστροφής είναι χρήσιμες για την εισαγωγή και εξαγωγή ακολουθιών bit σε λέξεις. Αν θέλουμε να ελέγξουμε όλα τα bit μιας λέξης, περιστρέφοντας τη λέξη κατά 1 bit τη φορά προς οποιαδήποτε κατεύθυνση κάθε bit περνάει από τη θέση του bit προσήμου, όπου μπορεί να ελεγχθεί εύκολα, και στο τέλος μετά τον έλεγχο όλων των bit η λέξη αποκαθίσταται στην αρχική της τιμή.

Όλες οι μηχανές των παραδειγμάτων μας διαθέτουν μια μεγάλη ποικιλία εντολών ολίσθησης και περιστροφής και προς τις δύο κατευθύνσεις. Μερικές από αυτές λαμβάνουν υπόψη και το bit κρατούμενου και μερικές όχι. Δεν έχει μεγάλη σημασία, ωστόσο, μια και οι μεταγωγιστές στην πράξη ποτέ δεν παράγουν κάποια από αυτές τις πράξεις εκτός της ολίσθησης προς τα αριστερά, ως ένα βελτιωμένο τρόπο πολλαπλασιασμού με δύναμη του δύο.

Ορισμένες δυαδικές πράξεις εμφανίζονται τόσο συχνά με συγκεκριμένους τελεστέους που οι μηχανές επιπέδου 2 μερικές φορές διαθέτουν μοναδιαίες εντολές για τη γρήγορη υλοποίησή τους. Η μεταφορά του μηδενός σε μια λέξη μνήμης ή καταχωρητή είναι ιδιαίτερα συνηθισμένη κατά την έναρξη ενός υπολογισμού. Η μεταφορά του μηδενός είναι, βέβαια, μια ειδική περίπτωση των γενικών εντολών μεταφοράς δεδομένων. Για λόγους αποδοτικότητας, διατίθεται συνήθως μια εντολή CLEAR, με μία μόνο διεύθυνση, τη διεύθυνση που πρέπει να μηδενιστεί.

Η πρόσθεση του 1 σε μια λέξη χρησιμοποιείται, επίσης, συχνά για μέτρηση. Μια μοναδιαία μορφή της εντολής πρόσθεσης είναι η πράξη αύξησης, που προσθέτει το 1. Η πράξη της άρνησης είναι ένα άλλο παράδειγμα. Η άρνηση του X ισοδυναμεί στην πραγματι-

κότητα με τον υπολογισμό του $0 - X$, μια δυαδική αφαίρεση, αλλά και πάλι για λόγους αποδοτικότητας, διατίθεται μερικές φορές ξεχωριστή εντολή άρνησης NEGATE.

Όλες οι μηχανές της Intel διαθέτουν εντολές INC και DEC για την πρόσθεση και αφαίρεση του 1. Ο 680x0 έχει κάπως πιο γενικές εντολές, τις ADDQ και SUBQ, που μπορούν να προσθέσουν ή να αφαιρέσουν μια σταθερά στην περιοχή από 1 ως 8.

5.4.4. Συγκρίσεις και Άλματα υπό Συνθήκη

Σχεδόν όλα τα προγράμματα πρέπει να είναι σε θέση να ελέγχουν τα δεδομένα τους και να αλλάζουν τη σειρά εκτέλεσης των εντολών με βάση τα αποτελέσματα. Ένα απλό παράδειγμα είναι η συνάρτηση τετραγωνικής ρίζας, \sqrt{x} . Αν ο αριθμός x είναι αρνητικός, η διαδικασία εμφανίζει μήνυμα σφάλματος· διαφορετικά, υπολογίζει την τετραγωνική ρίζα. Μια συνάρτηση *sqr* πρέπει να ελέγξει αν ο αριθμός x είναι αρνητικός ή όχι και στη συνέχεια να μεταβεί σε κάποιο σημείο, ανάλογα με το αποτέλεσμα του ελέγχου.

Μια συνηθισμένη μέθοδος για την υλοποίηση αυτής της διαδικασίας είναι η χρήση εντολών άλματος υπό συνθήκη (συνήν λέγονται εντολές διακλάδωσης υπό συνθήκη) που ελέγχουν κάποια συνθήκη και αν η συνθήκη ισχύει μεταβαίνουν σε κάποια συγκεκριμένη διεύθυνση μνήμης. Μερικές φορές κάποιο bit της εντολής παίρνει τιμή 1 ή 0, που σημαίνει ότι το άλμα θα γίνει αν ισχύει ή δεν ισχύει η συνθήκη, αντίστοιχα.

Η πιο συνηθισμένη συνθήκη που ελέγχεται είναι αν κάποιο συγκεκριμένο bit στη μηχανή έχει τιμή 0 ή όχι. Αν μια εντολή ελέγχει το bit προσήμου ενός αριθμού συμπληρώματος ως προς δύο και μεταβαίνει στην ετικέτα LABEL αν είναι 1, οι εντολές που αρχίζουν από τη θέση LABEL θα εκτελεστούν αν ο αριθμός ήταν αρνητικός, και οι εντολές που ακολουθούν μετά την εντολή άλματος υπό συνθήκη θα εκτελεστούν αν ήταν 0 ή θετικός. Αν γίνει ο ίδιος έλεγχος σ' έναν αριθμό συμπληρώματος ως προς ένα, το άλμα στη θέση LABEL θα γίνεται πάντα αν ο ελεγχόμενος αριθμός είναι μικρότερος ή ίσος με -1 και ποτέ αν είναι μεγαλύτερος ή ίσος με 1. Αν ο αριθμός είναι 0, το άλμα μπορεί να γίνει ή όχι, ανάλογα αν ο αριθμός είναι +0 ή -0. Αυτό προφανώς δεν είναι καθόλου ευχάριστο, επειδή από μαθηματική άποψη $+0 = -0$, και αποτελεί, μάλιστα, ένα από τα ισχυρότερα επιχειρήματα εναντίον της αριθμητικής συμπληρώματος ως προς ένα και υπέρ της αριθμητικής συμπληρώματος ως προς δύο.

Πολλές μηχανές χρησιμοποιούν κάποια bit για να υποδηλώνουν συγκεκριμένες συνθήκες. Για παράδειγμα, μπορεί να υπάρχει ένα bit υπερχείλισης που παίρνει τιμή 1 όποτε μια αριθμητική πράξη δίνει λάθος αποτέλεσμα. Ελέγχοντας αυτό το bit μπορούμε να εντοπίσουμε αν συνέβη υπερχείλιση στην προηγούμενη αριθμητική πράξη, οπότε αν συνέβη υπερχείλιση μπορεί να γίνει άλμα σε μια ρουτίνα χειρισμού σφαλμάτων. Ο 68000 διαθέτει μια ειδική, σύντομη εντολή, την TRAPV, που παγιδεύει την τιμή 1 του bit υπερχείλισης.

Παρόμοια, μερικοί επεξεργαστές διαθέτουν ένα bit κρατούμενου που παίρνει τιμή 1 όταν προκύπτει κρατούμενο από το αριστερότερο bit, για παράδειγμα, αν προστεθούν δύο αρνητικοί αριθμοί. Η εμφάνιση κρατούμενου από το αριστερότερο bit είναι αρκετά συνηθισμένη και δε θα πρέπει να συγχέεται με την περίπτωση της υπερχείλισης. Ο έλεγχος του bit κρατούμενου είναι απαραίτητος για αριθμητική πολλαπλής ακρίβειας.

Μερικές μηχανές διαθέτουν μια εντολή για τον έλεγχο του δεξιότερου bit μιας λέξης. Επιτρέπει στα προγράμματα να ελέγχουν αν ένας (θετικός) αριθμός είναι περιττός ή άρτιος σε μία εντολή.

Ο έλεγχος για μηδέν είναι σημαντικός σε βρόχους και πολλές άλλες περιπτώσεις. Αν όλες οι εντολές άλματος υπό συνθήκη έλεγχαν μόνο ένα bit, για να ελέγξουμε αν μια συγκεκριμένη λέξη ισούται με το 0, θα έπρεπε να ελέγξουμε κάθε bit χωριστά, για να βεβαιωθούμε ότι κανένα δεν έχει τιμή 1. Προς αποφυγήν αυτής της κατάστασης, πολλές μηχανές διαθέτουν μια εντολή που ελέγχει αν μια λέξη ισούται με μηδέν και ανάλογα εκτελεί άλμα. Βέβαια, αυτή η λύση απλώς μεταβιβάζει την ευθύνη στο επίπεδο 1. Στην πράξη, το υλικό συνήθως περιέχει έναν καταχωρητή στα bit του οποίου εκτελείται η πράξη OR, οπότε το αποτέλεσμα είναι ένα μόνο bit που προσδιορίζει αν ο καταχωρητής περιέχει bit με τιμή 1.

Η σύγκριση δύο λέξεων ή χαρακτήρων για να προσδιοριστεί αν είναι ίσοι ή ποιος είναι μεγαλύτερος, είναι, επίσης, σημαντική, για παράδειγμα στην ταξινόμηση. Για να εκτελεστεί αυτός ο έλεγχος, χρειάζονται τρεις διευθύνσεις, δύο για τα στοιχεία δεδομένων, και μία για τη διεύθυνση όπου θα γίνει η μετάβαση αν η συνθήκη είναι αληθής. Στους υπολογιστές των οποίων η μορφή εντολών επιτρέπει τρεις διευθύνσεις ανά εντολή δεν υπάρχει πρόβλημα, αλλά στους υπόλοιπους πρέπει να βρεθεί άλλος τρόπος για την εκτέλεση του ελέγχου.

Μια συνηθισμένη λύση είναι η χρήση μιας εντολής που εκτελεί σύγκριση και δίνει τιμή 1 σε ένα ή περισσότερα bit για να καταγράψει το αποτέλεσμα. Μια επόμενη εντολή μπορεί να ελέγξει τα bit συνθήκης και να μεταβεί σε κάποια διεύθυνση αν οι δύο συγκριθείσες τιμές ήταν ίσες, ή άνισες, ή αν η πρώτη ήταν μεγαλύτερη, κ.λπ. Και οι δύο μηχανές των παραδειγμάτων μας χρησιμοποιούν αυτή την προσέγγιση.

Μερικά λεπτά σημεία εμπλέκονται στη σύγκριση δύο αριθμών. Πρώτον, στις μηχανές συμπληρώματος ως προς ένα οι τιμές +0 και -0 είναι διαφορετικές· για παράδειγμα, σε μια μηχανή των 32 bit έχουμε:

```
00000000 00000000 00000000 00000000  +0 σε συμπλήρωμα ως προς ένα
11111111 11111111 11111111 11111111  -0 σε συμπλήρωμα ως προς ένα
```

Πρέπει να ληφθεί μια απόφαση από τους σχεδιαστές αν οι τιμές +0 και -0 είναι ίσες, και αν όχι, ποια είναι μεγαλύτερη. Ανεξάρτητα από το ποια θα είναι η απόφαση, υπάρχουν πειστικά επιχειρήματα ότι δε θα είναι η σωστή. Αν οι σχεδιαστές αποφασίσουν ότι $+0 = -0$, τότε το γεγονός ότι μια σύγκριση δίνει αποτέλεσμα ίσο δε σημαίνει ότι τα μοτίβα των bit των συγκρινόμενων δεδομένων είναι ίδια. Ας θεωρήσουμε μια μηχανή λέξεων των 32 bit όπου αποθηκεύονται τέσσερις χαρακτήρες των 8 bit. Αν σε μια λέξη αποθηκευτούν τέσσερις χαρακτήρες με κωδικό 0, η λέξη θα περιέχει 32 μηδέν. Αν σε μια λέξη αποθηκευτούν τέσσερις χαρακτήρες με κωδικό 255, η λέξη θα περιέχει 32 μονάδες. Αν αυτές οι δύο λέξεις συγκριθούν και προκύψουν ίσες, επειδή $+0 = -0$, ένα πρόγραμμα επεξεργασίας κειμένου μπορεί να βγάλει το λανθασμένο συμπέρασμα ότι οι δύο λέξεις περιέχουν τους ίδιους τέσσερις χαρακτήρες.

Αν, από την άλλη μεριά, το υλικό θεωρεί ότι οι τιμές +0 και -0 δεν είναι ίσες, συγκρίνοντας το αποτέλεσμα της πρόσθεσης του -1 στο +1 με την τιμή +0 μπορεί να προκύψει ότι είναι άνισα, επειδή η πρόσθεση του -1 στο +1 μπορεί να δώσει αποτέλεσμα -0. Δε χρειάζεται

να πούμε ότι μια τέτοια κατάσταση είναι ανεπιθύμητη. Οι μηχανές συμπληρώματος ως προς δύο, όπως των παραδειγμάτων μας, δεν έχουν προβλήματα με τις τιμές +0 και -0, αλλά οι μηχανές συμπληρώματος ως προς ένα έχουν. Σαν αποτέλεσμα, οι μηχανές συμπληρώματος ως προς ένα καταργούνται σταδιακά.

Ένα άλλο λεπτό σημείο σχετικά με τη σύγκριση αριθμών είναι η χρήση ή όχι προσήμου στους αριθμούς. Οι δυαδικοί αριθμοί τριών bit μπορεί να διαταχθούν κατά έναν από δύο τρόπους. Από το μικρότερο προς το μεγαλύτερο:

Μη προσημασμένοι	Προσημασμένοι
000	100 (μικρότερος)
001	101
010	110
011	111
100	000
101	001
110	010
111	011 (μεγαλύτερος)

Η αριστερή στήλη περιέχει τους θετικούς ακέραιους από 0 ως 7 σε αύξουσα σειρά. Η δεξιά στήλη περιέχει τους προσημασμένους ακέραιους συμπληρώματος ως προς δύο από -4 ως +3. Η απάντηση στο ερώτημα "Είναι το 011 μεγαλύτερο από το 100;" εξαρτάται από το αν οι αριθμοί θεωρούνται προσημασμένοι. Οι περισσότερες μηχανές, συμπεριλαμβανομένων των μηχανών της Intel και της Motorola, έχουν εντολές άλματος και για τις δύο σειρές αριθμών.

Οι μηχανές με λιγότερες από τρεις διευθύνσεις ανά εντολή μερικές φορές χειρίζονται τα άλματα υπό συνθήκη με μια εντολή που παραλείπει την επόμενη στη σειρά εντολή αν ισχύει η συνθήκη. Αυτή η εντολή συχνά είναι εντολή άλματος. Σε μερικές μηχανές μπορεί να παραλείπονται αρκετές εντολές, αντί για μία μόνο, όπου ο αριθμός των byte που παραλείπονται καθορίζεται στην ίδια την εντολή. Συνήθως πρόκειται για έναν αριθμό από -128 ως +127, ώστε να χωράει σε ένα byte. Ο 68000 διαθέτει αρκετές εντολές αυτού του τύπου.

Τα άλματα χωρίς συνθήκη αποτελούν μια ειδική περίπτωση αλμάτων υπό συνθήκη όπου η συνθήκη ισχύει πάντα.

5.4.5. Εντολές Κλήσης Διαδικασιών

Μια διαδικασία (procedure) είναι μια ομάδα εντολών που εκτελεί κάποια εργασία, και η οποία μπορεί να κληθεί από διάφορα σημεία ενός προγράμματος. Ο όρος υπορουτίνα (subroutine) χρησιμοποιείται συχνά αντί του όρου διαδικασία, ειδικότερα στα προγράμματα σε συμβολική γλώσσα. Όταν η διαδικασία ολοκληρώσει την εργασία της, πρέπει να επιστρέψει στις εντολές που βρίσκονται μετά την εντολή κλήσης. Επομένως, η διεύθυνση επιστροφής πρέπει να μεταδοθεί στη διαδικασία.

Η διεύθυνση επιστροφής μπορεί να τοποθετηθεί είτε στη μνήμη, είτε σε καταχωρητή, είτε στη στοίβα. Η χειρότερη λύση είναι να τοποθετηθεί σε μία σταθερή θέση της μνήμης. Σ' αυτή την περίπτωση, αν η διαδικασία καλεί μια άλλη διαδικασία, λόγω της δεύτερης κλήσης η διεύθυνση επιστροφής της πρώτης κλήσης θα χαθεί.

Μια μικρή βελτίωση έχουμε όταν η εντολή κλήσης της διαδικασίας αποθηκεύει τη διεύθυνση επιστροφής στην πρώτη λέξη της διαδικασίας, με την πρώτη εκτελέσιμη εντολή στη δεύτερη λέξη. Η διαδικασία μπορεί τότε να επιστρέψει με έμμεση μετάβαση στην πρώτη λέξη ή, αν το υλικό τοποθετεί τον κωδικό πράξης της εντολής άλματος στην πρώτη λέξη μαζί με τη διεύθυνση επιστροφής, με άμεση μετάβαση. Η διαδικασία μπορεί να καλεί άλλες διαδικασίες, επειδή κάθε διαδικασία έχει χώρο για μία διεύθυνση επιστροφής. Αν η διαδικασία καλεί τον εαυτό της, αυτό το σχήμα αποτυγχάνει, επειδή η πρώτη διεύθυνση επιστροφής θα καταστραφεί από τη δεύτερη κλήση. Η ικανότητα μιας διαδικασίας να καλεί τον εαυτό της λέγεται **αναδρομή** (recursion) και είναι ιδιαίτερα σημαντική και για τους θεωρητικούς και για τους πρακτικούς προγραμματιστές. Επίσης, αυτό το σχήμα αποτυγχάνει αν η διαδικασία A καλεί τη διαδικασία B, η διαδικασία B καλεί τη διαδικασία C, και η διαδικασία C καλεί τη διαδικασία A (έμμεση αναδρομή).

Μια ακόμη μεγαλύτερη βελτίωση επιτυγχάνεται αν η εντολή κλήσης της διαδικασίας τοποθετεί τη διεύθυνση επιστροφής σ' έναν καταχωρητή, αφήνοντας στη διαδικασία την ευθύνη αποθήκευσής της σε ασφαλές μέρος. Αν η διαδικασία είναι αναδρομική, θα πρέπει να τοποθετεί τη διεύθυνση επιστροφής σε διαφορετική θέση κάθε φορά που καλείται.

Η καλύτερη λύση είναι η εντολή κλήσης της διαδικασίας να τοποθετεί τη διεύθυνση επιστροφής σε μια στοίβα. Όταν η διαδικασία ολοκληρωθεί, αφαιρεί τη διεύθυνση επιστροφής από τη στοίβα και την τοποθετεί στο μετρητή προγράμματος. Αν αυτή η μορφή κλήσης διαδικασίας είναι διαθέσιμη, η αναδρομή δεν προκαλεί προβλήματα: η διεύθυνση επιστροφής θα αποθηκεύεται αυτόματα κατά τέτοιο τρόπο ώστε να αποφεύγεται η καταστροφή των προηγούμενων διευθύνσεων επιστροφής. Όλες οι μηχανές των παραδειγμάτων μας χρησιμοποιούν αυτή τη μέθοδο.

5.4.6. Έλεγχος Βρόχου

Η ανάγκη εκτέλεσης μιας ομάδας εντολών για ορισμένο αριθμό φορών εμφανίζεται συχνά κι έτσι μερικές μηχανές διαθέτουν εντολές που διευκολύνουν αυτή τη λειτουργία. Σε όλες τις μεθόδους χρησιμοποιείται ένας μετρητής που αυξάνεται ή μειώνεται κατά μια σταθερά, κάθε φορά που επαναλαμβάνεται ο βρόχος. Αυτός ο μετρητής ελέγχεται, επίσης, σε κάθε επανάληψη. Αν ισχύει κάποια συνθήκη, η εκτέλεση του βρόχου σταματάει.

Μια μέθοδος δίνει αρχική τιμή στο μετρητή έξω από το βρόχο και αρχίζει αμέσως μετά να εκτελεί τον κώδικα του βρόχου. Η τελευταία εντολή του βρόχου ενημερώνει το μετρητή και, αν η συνθήκη τερματισμού δεν είναι αληθής, μεταβαίνει ξανά στην πρώτη εντολή του βρόχου. Διαφορετικά, η εκτέλεση του βρόχου σταματάει και εκτελείται η πρώτη εντολή μετά το βρόχο. Σ' αυτή τη μορφή βρόχου ο έλεγχος της συνθήκης γίνεται στο τέλος, όπως φαίνεται στην Εικόνα 5-40(α).

```

i := 1;
1: {πρώτη εντολή}
   {δεύτερη εντολή}
   .
   .
   {τελευταία εντολή}
   i := i + 1;
   If i <= n then goto 1;
2: {πρώτη εντολή μετά το βρόχο}

```

(α)

Εικόνα 5-40. (α) Έλεγχος στο τέλος του βρόχου. (β) Έλεγχος στην αρχή του βρόχου.

Ο βρόχος με έλεγχο στο τέλος έχει την ιδιότητα να εκτελείται τουλάχιστον μία φορά, ακόμη κι αν ο αριθμός n είναι μικρότερος ή ίσος με το 0. Ας θεωρήσουμε, για παράδειγμα, ένα πρόγραμμα διαχείρισης αρχείων προσωπικού μιας εταιρίας. Σε κάποιο σημείο, το πρόγραμμα διαβάζει πληροφορίες για ένα συγκεκριμένο υπάλληλο. Διαβάζει τον αριθμό n , που αντιστοιχεί στο πλήθος των παιδιών του υπαλλήλου, και εκτελεί ένα βρόχο n φορές, μία για κάθε παιδί, διαβάζοντας το όνομα του παιδιού, το φύλο, και την ημερομηνία γέννησης, ώστε η εταιρία να μπορεί να στείλει ένα δώρο γενεθλίων ως μέρος των παροχών της προς τους υπαλλήλους. Αν ο υπάλληλος δεν έχει παιδιά, ο αριθμός n θα ισούται με 0, αλλά ο βρόχος θα εκτελεστεί μία φορά στέλνοντας δώρα και δίνοντας λανθασμένα αποτελέσματα.

Στην Εικόνα 5-40(β) παρουσιάζεται ένας άλλος τρόπος ελέγχου που δουλεύει σωστά ακόμη και για τιμές του n μικρότερες ή ίσες του 0. Παρατηρήστε ότι ο έλεγχος διαφέρει στις δύο περιπτώσεις, ώστε αν η αύξηση του μετρητή και ο έλεγχος γίνονται από μία μόνο εντολή επιπέδου 2, οι σχεδιαστές είναι υποχρεωμένοι να επιλέξουν μία από τις δύο μεθόδους.

Ας θεωρήσουμε τον κώδικα που θα πρέπει να δημιουργήσει ο μεταγλωττιστής για την παρακάτω πρόταση Pascal

```
for i := 1 to n do begin ... end
```

Αν ο μεταγλωττιστής δεν έχει κάποια πληροφορία για τον αριθμό n , θα πρέπει να χρησιμοποιήσει τη μέθοδο της Εικόνας 5-40(β) για να χειριστεί σωστά την περίπτωση $n <= 0$. Αν, ωστόσο, μπορεί να προσδιορίσει ότι $n > 0$, για παράδειγμα, βλέποντας πού έχει εκχωρηθεί το n , μπορεί να χρησιμοποιήσει τον καλύτερο κώδικα της Εικόνας 5-40(α). Η βασική έκδοση της γλώσσας FORTRAN καθόριζε ότι όλοι οι βρόχοι θα πρέπει να εκτελούνται τουλάχιστον μία φορά, ώστε να παράγεται πάντα ο πιο αποτελεσματικός κώδικας της Εικόνας 5-40(α). Το 1977, αυτό το μειονέκτημα διορθώθηκε όταν ακόμη και οι οπαδοί της FORTRAN κατάλαβαν ότι η χρήση μιας εντολής βρόχου με παράξενη σημειολογία δεν ήταν καλή ιδέα, έστω κι αν εξοικονομούσε μια εντολή άλματος σε κάθε εκτέλεση του βρόχου.

```

i := 1;
1: if i > n then goto 2;
   {πρώτη εντολή}
   {δεύτερη εντολή}
   .
   .
   {τελευταία εντολή}
   i := i + 1;
   goto 1;
2: {πρώτη εντολή μετά το βρόχο}

```

(β)

Όλες οι CPU της Intel εκτελούν ένα βρόχο ορισμένες φορές χρησιμοποιώντας την εντολή LOOP, που μειώνει τους καταχωρητές CX/ECX κατά 1, και μεταβαίνει σε μια συγκεκριμένη ετικέτα αν το αποτέλεσμα δεν είναι μηδέν. Αν είναι μηδέν, ο βρόχος ολοκληρώνεται και η εκτέλεση συνεχίζεται με την επόμενη εντολή.

Ο 680x0 διαθέτει μια κάπως πιο γενική εντολή που ελέγχει πρώτα αν ισχύει μια συγκεκριμένη συνθήκη, οπότε βγαίνει από το βρόχο. Αν η συνθήκη δεν ισχύει, μειώνεται ένας καταχωρητής D. Αν το αποτέλεσμα είναι 0 ή μεγαλύτερο, ο βρόχος επαναλαμβάνεται διαφορετικά, ο βρόχος ολοκληρώνεται και εκτελείται η εντολή που ακολουθεί.

5.4.7. Είσοδος/Εξόδος

Καμμία άλλη ομάδα εντολών δεν διαφέρει τόσο πολύ από μηχανή σε μηχανή όσο οι εντολές E/E. Υπάρχουν τέσσερις διαφορετικές μέθοδοι E/E, οι εξής:

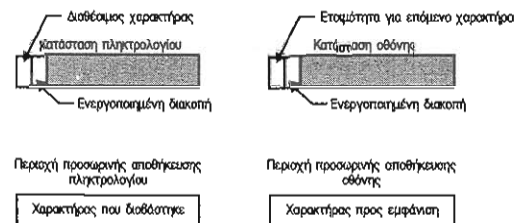
1. Προγραμματιζόμενη E/E με αναμονή λόγω απασχόλησης.
2. E/E με διακοπές.
3. E/E DMA.
4. E/E με κανάλια δεδομένων.

Θα εξετάσουμε με τη σειρά κάθε μία από αυτές.

Η πιο απλή μέθοδος E/E είναι η προγραμματιζόμενη E/E (programmed I/O), που χρησιμοποιείται σε μερικούς υπολογιστές χαμηλής απόδοσης. Αυτοί οι μικροϋπολογιστές έχουν μόνο μία εντολή εισόδου και μία εντολή εξόδου. Κάθε μία από αυτές τις εντολές επιλέγει μία από τις συσκευές E/E. Ένας μόνο χαρακτήρας μεταφέρεται μεταξύ ενός καθορισμένου καταχωρητή του επεξεργαστή και της επιλεγμένης συσκευής E/E. Ο επεξεργαστής πρέπει να εκτελέσει μια άμεση εντολή για κάθε ξεχωριστό χαρακτήρα που διαβάζεται ή γράφεται.

Ένα απλό παράδειγμα αυτής της μεθόδου είναι ένα τερματικό με τέσσερις καταχωρητές του 1 byte, όπως φαίνεται στην Εικόνα 5-41. Δύο καταχωρητές χρησιμοποιούνται για είσοδο, ένας καταχωρητής κατάστασης και ένας καταχωρητής δεδομένων, και δύο για έξοδο, επίσης κατάστασης και δεδομένων. Καθένας έχει μοναδική διεύθυνση. Αν χρησιμοποιείται E/E χαρτογράφησης μνήμης, και οι τέσσερις καταχωρητές αποτελούν τμήμα του διαστήματος διευθύνσεων μνήμης του υπολογιστή, και μπορούν να διαβαστούν και να γραφούν με συνηθισμένες εντολές. Διαφορετικά, για την εγγραφή και την ανάγνωση των καταχωρητών παρέχονται ειδικές εντολές, όπως οι IN και OUT, για παράδειγμα. Και στις δύο περιπτώσεις, η E/E εκτελείται μεταφέροντας δεδομένα και πληροφορίες κατάστασης μεταξύ της CPU και αυτών των καταχωρητών.

Ο καταχωρητής κατάστασης πληκτρολογίου έχει δύο bit που χρησιμοποιούνται και 6 bit που δε χρησιμοποιούνται. Το αριστερότερο bit (7) παίρνει τιμή 1 από το υλικό όποτε λαμβάνεται ένας χαρακτήρας. Αν το λογισμικό έχει προηγουμένως δώσει τιμή 1 στο bit 6, παράγεται μια διακοπή, διαφορετικά δεν παράγεται διακοπή (θα μιλήσουμε για τις διακοπές σύντομα).



Εικόνα 5-41. Καταχωρητές συσκευής ενός απλού τερματικού.

Κατά τη χρήση προγραμματιζόμενης Ε/Ε, για την ανάκτηση εισόδου, η CPU κανονικά εκτελεί συνεχώς ένα βρόχο διαβάζοντας τον καταχωρητή κατάστασης πληκτρολογίου και περιμένοντας το bit 7 να πάρει τιμή 1. Όταν συμβεί αυτό, το λογισμικό διαβάζει τον καταχωρητή προσωρινής αποθήκευσης για να ανακτήσει το χαρακτήρα. Η ανάγνωση του καταχωρητή δεδομένων πληκτρολογίου προκαλεί το μηδενισμό του bit ΔΙΑΘΕΣΙΜΟΥ ΧΑΡΑΚΤΗΡΑ (AVAILABLE CHARACTER).

Η έξοδος λειτουργεί παρόμοια. Για να γράψουμε ένα χαρακτήρα στην οθόνη, το λογισμικό πρώτα ελέγχει αν το bit ΕΤΟΙΜΟΤΗΤΑΣ (READY) του καταχωρητή κατάστασης οθόνης έχει τιμή 1. Αν όχι, επαναλαμβάνει ένα βρόχο μέχρι το bit να πάρει τιμή 1, υποδηλώνοντας ότι η συσκευή είναι έτοιμη να δεχτεί ένα χαρακτήρα. Αμέσως μόλις το τερματικό βρεθεί σε κατάσταση ετοιμότητας, το λογισμικό γράφει ένα χαρακτήρα στον καταχωρητή προσωρινής αποθήκευσης οθόνης, οπότε ο χαρακτήρας μεταφέρεται στην οθόνη και το bit ΕΤΟΙΜΟΤΗΤΑΣ του καταχωρητή κατάστασης οθόνης μηδενίζεται. Όταν ο χαρακτήρας εμφανιστεί στην οθόνη και το τερματικό ετοιμαστεί να χειριστεί το δεύτερο χαρακτήρα, ο ελεγκτής εκχωρεί ξανά την τιμή 1 αυτομάτως στο bit ΕΤΟΙΜΟΤΗΤΑΣ.

Ένα παράδειγμα προγραμματιζόμενης Ε/Ε είναι η διαδικασία Pascal της Εικόνας 5-42. Αυτή η διαδικασία καλείται με δύο παραμέτρους: έναν πίνακα χαρακτήρων για έξοδο, και το πλήθος των χαρακτήρων του πίνακα, μέχρι 1K. Το σώμα της διαδικασίας είναι ένας βρόχος που εξάγει χαρακτήρες ένα προς ένα. Για κάθε χαρακτήρα, η CPU πρέπει πρώτα να περιμένει μέχρι να ετοιμαστεί η συσκευή, και στη συνέχεια ο χαρακτήρας εξάγεται. Οι διαδικασίες *in* και *out* συνήθως είναι διαδικασίες σε συμβολική γλώσσα για την ανάγνωση και εγγραφή των καταχωρητών της συσκευής που καθορίζονται από την πρώτη παράμετρο από ή προς τη μεταβλητή που καθορίζεται από τη δεύτερη παράμετρο. Η διαίρεση με το 128 μας απαλλάσσει από τα 7 bit χαμηλής τάξης, αφήνοντας ως bit ΕΤΟΙΜΟΤΗΤΑΣ το bit 0.

Το κύριο μειονέκτημα της προγραμματιζόμενης Ε/Ε είναι ότι η CPU καταναλώνει τον περισσότερο χρόνο της περιμένοντας σ' ένα βρόχο τη συσκευή να βρεθεί σε κατάσταση ετοιμότητας. Αυτή η προσέγγιση λέγεται *αναμονή λόγω απασχόλησης* (busy waiting). Αν η CPU δεν έχει τίποτα άλλο να κάνει (για παράδειγμα, η CPU ενός πλυντηρίου), η μέθοδος αναμονής λόγω απασχόλησης επαρκεί. Ωστόσο, αν υπάρχουν κι άλλες εργασίες, όπως η εκτέλεση άλλων προγραμμάτων, η μέθοδος αυτή αποτελεί σπατάλη χρόνου, οπότε απαιτείται κάποια άλλη μέθοδος.

```

const size = 1023;

type buffer = array [0..size] of char;

procedure OutputBuf(b: buffer; count: integer);
{Έξοδος ενός μπλοκ δεδομένων προς το τερματικό.}
var status, i, ready: integer;
begin
  for i:=0 to count do
  begin
    repeat                                     {Για κάθε χαρακτήρα, εκτέλεση ενός βρόχου και}
    until ready = 1;                           {έξοδος 1 χαρακτήρα}
    in(DisplayStatusReg, status); {ανάκτηση του καταχωρητή κατάστασης}
    ready := status div 128;      {ολίσθηση των bit 0-6 εκτός}
    out(DisplayBufferReg, b[i])  {έξοδος ενός χαρακτήρα}
  end
end;

```

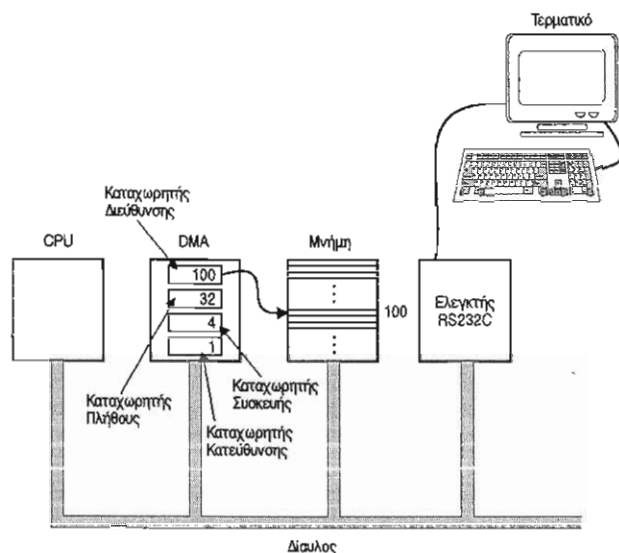
Εικόνα 5-42. Παράδειγμα προγραμματιζόμενης Ε/Ε.

Ένας τρόπος αποφυγής της αναμονής λόγω απασχόλησης είναι η χρήση διακοπών: η CPU δίνει εντολή στη συσκευή Ε/Ε να ξεκινήσει τη λειτουργία της, και όταν αυτή ολοκληρωθεί, η συσκευή παράγει ένα σήμα διακοπής. Μπορούμε να εξηγήσουμε αυτή τη διαδικασία με τη βοήθεια της Εικόνας 5-41. Εκχωρώντας την τιμή 1 στο bit ΕΝΕΡΓΟΠΟΙΗΣΗΣ ΔΙΑΚΟΠΗΣ (INTERRUPT ENABLE) ενός καταχωρητή συσκευής, το λογισμικό μπορεί να απαιτήσει από το υλικό την αποστολή ενός σήματος όταν ολοκληρωθεί η λειτουργία Ε/Ε. Θα εξετάσουμε τις διακοπές λεπτομερώς στη συνέχεια του κεφαλαίου, όταν μιλήσουμε για τη ροή του ελέγχου.

Αξίζει να αναφέρουμε ότι σε πολλούς υπολογιστές το σήμα διακοπής παράγεται εφαρμόζοντας τη σύνδεση AND στα bit ΕΝΕΡΓΟΠΟΙΗΣΗΣ ΔΙΑΚΟΠΗΣ και ΕΤΟΙΜΟΤΗΤΑΣ. Αν το λογισμικό ενεργοποιεί πρώτα τις διακοπές (πριν ξεκινήσει τη λειτουργία Ε/Ε), θα συμβεί αυτόματα μια διακοπή, επειδή το bit ΕΤΟΙΜΟΤΗΤΑΣ θα έχει τιμή 1. Έτσι, ίσως πρέπει πρώτα να ξεκινάει η συσκευή, και αμέσως μετά να ενεργοποιούνται οι διακοπές. Η εγγραφή ενός byte στον καταχωρητή κατάστασης δεν αλλάζει το bit ΕΤΟΙΜΟΤΗΤΑΣ, που είναι μόνο-για-ανάγνωση.

Αν και η λειτουργία Ε/Ε με διακοπές αποτελεί μεγάλη βελτίωση σε σχέση με την προγραμματιζόμενη Ε/Ε, απέχει πολύ από το να είναι τέλεια. Το πρόβλημα είναι ότι απαιτείται μια διακοπή για κάθε μεταδιδόμενο χαρακτήρα. Η επεξεργασία μιας διακοπής είναι δαπανηρή. Χρειάζεται κάποιος τρόπος που να μας απαλλάσσει από τις περισσότερες διακοπές.

Η λύση είναι να επιστρέψουμε στην προγραμματιζόμενη Ε/Ε, αλλά να αναθέσουμε σε κάποιον άλλο την υλοποίησή της. (Η λύση σε πολλά προβλήματα βρίσκεται στην ανάθεσή τους σε κάποιον άλλο.) Στην Εικόνα 5-43 παρουσιάζεται η νέα μέθοδος. Έχει προστεθεί στο σύστημα ένα νέο τσιπ, ο ελεγκτής DMA (Direct Memory Access — Άμεση Προσπέλαση Μνήμης), που έχει άμεση πρόσβαση στο δίαυλο.



Εικόνα 5-43. Ένα σύστημα με ελεγκτή DMA.

Το τσιπ DMA έχει (τουλάχιστον) τέσσερις καταχωρητές στο εσωτερικό του, οι οποίοι μπορούν να φορτωθούν από λογισμικό που εκτελείται στη CPU. Ο πρώτος περιέχει τη διεύθυνση μνήμης που θα διαβαστεί ή θα γραφεί. Ο δεύτερος περιέχει το πλήθος των byte (ή λέξεων) που θα μεταφερθούν. Ο τρίτος καθορίζει τον αριθμό συσκευής ή τη διεύθυνση διαστήματος E/E που θα χρησιμοποιηθεί. Ο τέταρτος καθορίζει αν τα δεδομένα θα διαβαστούν ή θα γραφούν στη συσκευή E/E.

Για να γράψει ένα μπλοκ μεγέθους 32 byte από τη διεύθυνση μνήμης 100 σ' ένα τερματικό (ας πούμε, τη συσκευή 4), η CPU γράφει τους αριθμούς 32, 100, και 4 στους πρώτους τρεις καταχωρητές DMA, και μετά τον κωδικό για ΕΓΓΡΑΦΗ (ας πούμε, το 1) στον τέταρτο, όπως φαίνεται στην Εικόνα 5-43. Μόλις αποδοθούν οι αρχικές τιμές, ο ελεγκτής DMA πραγματοποιεί μια αίτηση E/E προς τη συσκευή 4, για να γράψει σ' αυτή το byte. Μετά των ολοκλήρωση και των δύο αυτών λειτουργιών, ο ελεγκτής DMA αυξάνει τον καταχωρητή διεύθυνσης κατά 1 και μειώνει τον καταχωρητή πλήθους κατά 1. Αν η τιμή του καταχωρητή πλήθους εξακολουθεί να είναι μεγαλύτερη από 0, διαβάζεται κι άλλο byte από τη μνήμη και γράφεται στη συσκευή.

Όταν η τιμή του καταχωρητή πλήθους γίνει τελικά ίση με το 0, ο ελεγκτής DMA σταματάει τη μεταφορά δεδομένων και ενεργοποιεί τη γραμμή διακοπών στο τσιπ της CPU. Με τη μέθοδο DMA, η CPU το μόνο που κάνει είναι να εκχωρήσει αρχικές τιμές σε μερικούς καταχωρητές. Μετά, είναι ελεύθερη να εκτελέσει άλλες εργασίες μέχρι να ολοκληρωθεί η

μεταφορά, οπότε λαμβάνει ένα σήμα διακοπής από τον ελεγκτή DMA. Μερικοί ελεγκτές DMA διαθέτουν δύο, ή τρεις, ή περισσότερες ομάδες καταχωρητών, οπότε μπορεί να ελέγχουν πολλές ταυτόχρονες μεταφορές.

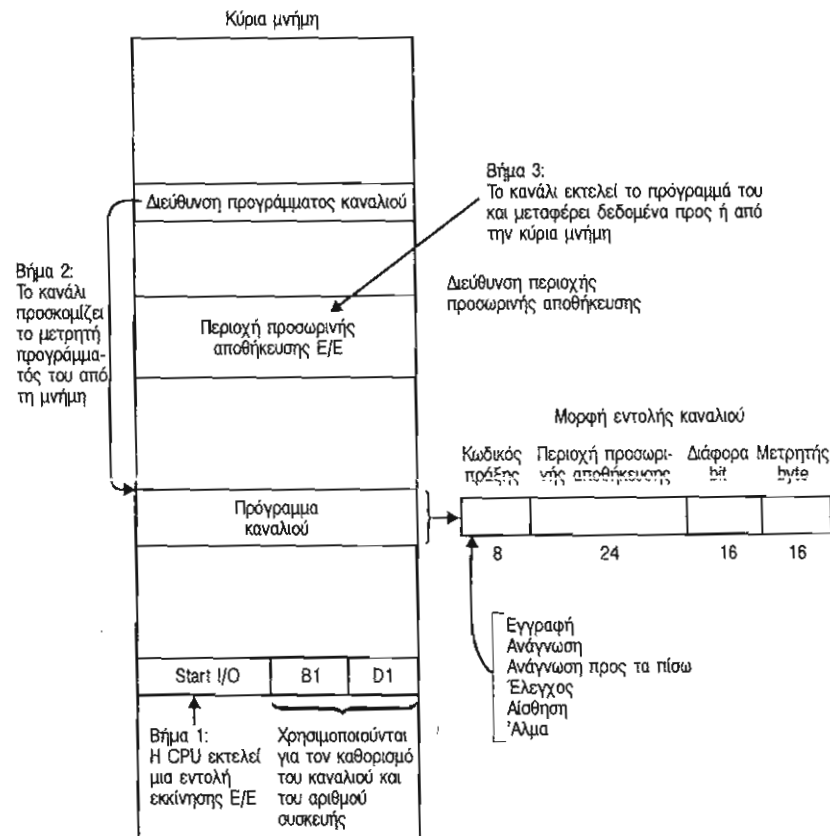
Αν και η μέθοδος DMA απελευθερώνει σε μεγάλο βαθμό τη CPU από την ευθύνη της λειτουργίας E/E, δεν είναι τελείως απαλλαγμένη. Αν μια συσκευή υψηλής ταχύτητας, όπως ο σκληρός δίσκος, ελέγχεται από ελεγκτή DMA, θα χρειάζονται πολλοί κύκλοι διαύλου, και για τις αναφορές στη μνήμη και για τις αναφορές στη συσκευή. Κατά τη διάρκεια αυτών των κύκλων, η CPU θα πρέπει να περιμένει (ο ελεγκτής DMA έχει πάντα μεγαλύτερη προτεραιότητα στη χρήση του διαύλου από τη CPU επειδή οι συσκευές E/E συνήθως δεν ανέχονται καθυστερήσεις). Η διαδικασία κατά την οποία ο ελεγκτής DMA αφαιρεί κύκλους διαύλου από τη CPU λέγεται **αφαίρεση κύκλων** (cycle stealing). Ωστόσο, το κέρδος από το γεγονός ότι δε χρειάζεται πλέον μία διακοπή για κάθε μεταφορά χαρακτήρα (ή λέξης) είναι πολύ μεγαλύτερο από τις απώλειες λόγω αφαίρεσης κύκλων. Η μέθοδος DMA είναι η συνήθης μέθοδος υλοποίησης της λειτουργίας E/E σε όλους τους προσωπικούς υπολογιστές και mini-υπολογιστές.

Στα μεγάλα συστήματα υπολογιστών, ωστόσο, η κατάσταση είναι διαφορετική. Σ' αυτές τις μηχανές συνήθως υπάρχει τόσο μεγάλος φόρτος E/E ώστε η αφαίρεση κύκλων θα προκαλούσε κορεσμό του διαύλου, και ακόμη και με μία διακοπή ανά μεταφερόμενο μπλοκ, πάρα πολύς χρόνος θα αφιερωνόταν στο χειρισμό διακοπών. Η λύση στην περίπτωση αυτή είναι η προσθήκη στην αρχιτεκτονική ειδικών επεξεργασιών E/E, που λέγονται **κανάλια δεδομένων** (data channels), όπως φαίνεται στην Εικόνα 2-19.

Ένα κανάλι είναι στην πραγματικότητα ένας εξειδικευμένος υπολογιστής. Μπορούμε να του δώσουμε ένα πρόγραμμα για εκτέλεση, και θα ξεκινήσει να εκτελεί το πρόγραμμα χωρίς καμία βοήθεια από την κύρια CPU. Όταν ολοκληρωθεί η εκτέλεση του προγράμματος, το κανάλι διακόπτει τη CPU. Μία και ένα πρόγραμμα καναλιού μπορεί να είναι αρκετά πολύπλοκο, και ασχολείται με τη μεταφορά πολλών μπλοκ δεδομένων, απαιτούνται λιγότερες διακοπές απ' ό,τι στις απλές μεταφορές DMA.

Επειδή ούτε η Intel ούτε η Motorola χρησιμοποιούν κανάλια δεδομένων, θα χρησιμοποιήσουμε ως παράδειγμα τη δομή E/E των μεγάλων συστημάτων υπολογιστών της IBM. Υπάρχουν δύο τύποι καναλιών. Ένα **κανάλι επιλογών** (selector channel) ελέγχει τις συσκευές υψηλής ταχύτητας, όπως οι δίσκοι. Λόγω της μεγάλης ταχύτητας μεταφοράς δεδομένων από αυτές τις συσκευές, ένα κανάλι επιλογών μπορεί να χειριστεί μόνο μία μεταφορά τη φορά. Αντίθετα, ένα **κανάλι πολύπλεξης** (multiplexor channel) μπορεί να χειριστεί ταυτόχρονα πολλές συσκευές χαμηλής ταχύτητας, όπως τερματικά.

Για την εκτέλεση της λειτουργίας E/E σ' έναν υπολογιστή με κανάλια δεδομένων, η CPU δημιουργεί πρώτα ένα πρόγραμμα για το κανάλι και το αποθηκεύει στην κύρια μνήμη. Στη συνέχεια, εκτελεί μια εντολή START I/O (Εκκίνηση E/E) καθορίζοντας το κανάλι και τη συσκευή E/E. Το κανάλι τότε ανακτά τη διεύθυνση του προγράμματός του από μια συγκεκριμένη θέση της μνήμης, τοποθετεί αυτή τη διεύθυνση στο μετρητή προγράμματός του, και αρχίζει την εκτέλεση του προγράμματος καναλιού. Οι διάφορες λέξεις της μνήμης που εμπλέκονται στη διαδικασία παρουσιάζονται στην Εικόνα 5-44 για ένα τυπικό μεγάλο σύστημα υπολογιστών.



Εικόνα 5-44. Τυπικά βήματα υλοποίησης της λειτουργίας E/E χρησιμοποιώντας κανάλι δεδομένων.

Ένα πρόγραμμα καναλιού αποτελείται από μία ή περισσότερες εντολές των 64 bit για το κανάλι. Κάθε εντολή περιέχει έναν κωδικό πράξης των 8 bit που καθορίζει ποια ενέργεια θα εκτελεστεί. Μεταξύ αυτών περιλαμβάνονται οι εντολές READ (Ανάγνωση), WRITE (Εγγραφή), READ BACKWARD (Ανάγνωση προς τα πίσω — για παράδειγμα, τύλιγμα μαγνητικών ταινιών), CONTROL (Έλεγχος — για παράδειγμα, εκκίνηση μοτέρ), SENSE (Αίσθηση — για παράδειγμα, έλεγχος τέλους αρχείου), CONDITIONAL BRANCH (Διακλάδωση υπό Συνθή-

κη). Οι εντολές καναλιού περιέχουν, επίσης, μια διεύθυνση περιοχής προσωρινής αποθήκευσης των 24 bit που καθορίζει από πού θα διαβαστούν ή πού θα γραφούν τα δεδομένα, ένα μετρητή που καθορίζει πόσα byte θα μεταφερθούν, και μερικά bit σημαιών. Τα bit σημαιών καθορίζουν στοιχεία όπως "όχι μετάδοση δεδομένων" (για παράλειψη μιας εγγραφής σε ταινίες) και "τερματισμός του καναλιού μετά την ολοκλήρωση αυτής της εντολής".

Εκτός από την εντολή START I/O, η CPU διαθέτει μερικές ακόμη εντολές E/E. Η εντολή HALT I/O (Στάση λειτουργίας E/E) διακόπτει απότομα όλες τις δραστηριότητες στο επιλεγμένο κανάλι. Οι εντολές TEST I/O (Έλεγχος λειτουργίας E/E) και TEST CHANNEL (Έλεγχος καναλιού) χρησιμοποιούνται για τον προσδιορισμό της τρέχουσας κατάστασης της δραστηριότητας E/E. Υπάρχουν, επίσης, μερικές ακόμη δευτερεύουσες εντολές E/E.

Όλες οι CPU της Intel διαθέτουν άμεσες εντολές E/E για την ανάγνωση ή εγγραφή byte, λέξεων, ή μεγάλων λέξεων. Αυτές οι εντολές καθορίζουν τον αριθμό θύρας E/E, είτε άμεσα, ως πεδίο μέσα στην εντολή, είτε έμμεσα, χρησιμοποιώντας τον καταχωρητή DX. Επιπλέον, βέβαια, τα τσιπ DMA χρησιμοποιούνται συχνά για την απαλλαγή της CPU από το βάρος της λειτουργίας E/E.

Κανένα από τα τσιπ της Motorola δεν έχει εντολές E/E. Οι καταχωρητές συσκευών E/E προσπαλάζονται μέσω χαρτογράφησης μνήμης. Και εδώ χρησιμοποιείται ευρέως η μέθοδος DMA.