Πανεπιστήμιο Θεσσαλίας
Τμήμα Πληροφορικής

Οργάνωση Η/Υ

14 Νοεμβρίου 2018

Α. Πρώτη Σειρά Ασκήσεων
παράδοση: 12 Δεκεμβρίου 3μμ

Άσκηση 1

Θεωρήστε τις ΜΕΔ της αρχιτεκτονικής MIPS καλωδιωμένης λογικής για κύκλο εντολής τόσο απλού, όσο και πολλαπλών κύκλων μηχανής. Περιγράψτε την *πλήρη* ροή πληροφορίας και στις δύο περιπτώσεις υλοποίησης – απλού και πολλαπλών κύκλων μηχανής – για την εντολή:

```
bne $12, $4, -100
```

όπου η σταθερά -100 αποτελεί τη μετατόπιση εντολών της σχετικής διευθυνσιοδότησης.
Ειδικότερα, να αναφέρετε όλες τις τιμές των σημάτων ελέγχου που παράγονται, καθώς και τις μικρολειτουργίες που εκτελούνται σε κάθε υπομονάδα της ΜΕΔ, με τη σειρά που αυτές εκτελούνται, είτε είναι χρήσιμες για τη συγκεκριμένη εντολή είτε όχι – γεγονός το οποίο ζητείται να αναφέρετε. Ακόμη, να γράψετε τις τιμές πληροφορίας που φτάνουν σε κάθε υπομονάδα και κάθε πολυπλέκτη. Τέλος, για την περίπτωση πολλαπλών κύκλων μηχανής, να αναφέρετε τις εγγραφές των καταχωρητών A, B, C, IR και DR που συμβαίνουν, καθώς και τις τιμές που εγγράφονται.
Θεωρήστε ότι στην αρχή του κύκλου εντολής οι καταχωρητές $12 και $4 περιέχουν τις τιμές 8 και 0 αντίστοιχα, ενώ ο PC περιέχει τη διεύθυνση 0x1480a008.
Πώς θα εξελισσόταν η εκτέλεση της παραπάνω εντολής, εάν ο καταχωρητής $4 στην αρχή του κύκλου εντολής περιείχε την τιμή 8;

Άσκηση 2

Θεωρήστε τη ΜΕΔ της αρχιτεκτονικής MIPS καλωδιωμένης λογικής για κύκλο εντολής τόσο απλού, όσο και πολλαπλών κύκλων μηχανής. Υποθέστε ότι θέλουμε να επεκτείνουμε το σύνολο εντολών, ώστε να περιλαμβάνει κάποιες πιο απλές, αλλά και κάποιες πιο ισχυρές εντολές, χρήσιμες για πολλές εφαρμογές:

Α. Εντολές προσπέλασης μνήμης χωρίς μετατόπιση:

```
lw $rt,($rs)
sw $rt,($rs)
```

όπου η τελική διεύθυνση προσπέλασης προέρχεται κατ' ευθείαν από τον καταχωρητή $rs.

Β. Εντολές προσπέλασης μνήμης με διπλά έμμεση διευθυνσιοδότηση χωρίς μετατόπιση:

```
lwd $rt,(($rs))
swd $rt,(($rs))
```

όπου η τελική διεύθυνση προσπέλασης είναι το περιεχόμενο της θέσης μνήμης με διεύθυνση που προέρχεται από τον καταχωρητή $rs. Για την πρώτη εντολή, το αποτέλεσμα της φόρτωσης αποθηκεύεται στον καταχωρητή $rt, ενώ για τη δεύτερη εντολή, το περιεχόμενο του καταχωρητή $rt αποθηκεύεται στη μνήμη.

Γ. Εντολή αντιγραφής μεταξύ θέσεων μνήμης:

```
move ($rt),off($rs)
```

όπου η τελική διεύθυνση προσπέλασης ανάγνωσης υπολογίζεται όπως ακριβώς στην κλασική αρχιτεκτονική MIPS, ενώ η τελική διεύθυνση προσπέλασης εγγραφής προέρχεται κατ' ευθείαν από τον καταχωρητή $rt.

Δ. Εντολές έμμεσης διακλάδωσης με σύνδεση:

```
beqzmal $rd,$rt,($rs)
bnezmal $rd,$rt,($rs)
```

οι οποίες εκτελούν άλμα με συνθήκη τη μηδενική ή μη μηδενική τιμή του καταχωρητή $rt, με προορισμό διεύθυνση που περιέχεται στη μνήμη, σε διεύθυνση που προέρχεται κατ' ευθείαν από τον καταχωρητή $rs, και με σύνδεση στον καταχωρητή $rd.

Χωρίς να εισάγετε νέες υπομονάδες στη ΜΕΔ – πλην πολυπλεκτών, εξηγήστε εάν και πώς μπορείτε να υποστηρίξετε τις παραπάνω εντολές, τόσο στην περίπτωση απλού, όσο και στην περίπτωση πολλαπλών κύκλων μηχανής για κάθε κύκλο εντολής. Πιο συγκεκριμένα, τι προσθήκες χρειάζεστε στη ΜΕΔ για την υποστήριξη των εντολών αυτών; Χρειάζεστε κάποιους νέους δρόμους μεταφοράς πληροφορίας; Χρειάζεστε κάποια νέα σήματα ελέγχου;
Οι όποιες τροποποιήσεις θα πρέπει να ελαχιστοποιούν τη διάρκεια του κύκλου μηχανής στην πρώτη, και τον αριθμό κύκλων μηχανής για τις εμπλεκόμενες εντολές στη δεύτερη περίπτωση. Ειδικά για την περίπτωση πολλαπλών κύκλων, δεν θα πρέπει να αυξάνεται η διάρκεια του κύκλου μηχανής.
Στην περίπτωση των πολλαπλών κύκλων μηχανής, να δώσετε την απαραίτητη προσθήκη στη μηχανή καταστάσεων που να περιγράφει την εκτέλεση των εντολών.

Να απαντήσετε ανεξάρτητα για καθένα από τα ερωτήματα Α-Δ.

Άσκηση 3

Θεωρήστε τη ΜΕΔ της αρχιτεκτονικής MIPS καλωδιωμένης λογικής για κύκλο εντολής τόσο απλού, όσο και πολλαπλών κύκλων μηχανής. Επειδή κρίθηκε ότι τα άμεσα τελούμενα και μετατοπίσεις εύρους 16 bits δεν καλύπτουν τις ανάγκες των εφαρμογών, προτάθηκε μια τροποποίηση της αρχιτεκτονικής συνόλου εντολών, όπου οι εντολές με άμεσο τελούμενο ή μετατόπιση γίνονται δύο λέξεων, με την δεύτερη να περιέχει το άμεσο τελούμενο ή τη μετατόπιση με εύρος 32 bits, και την πρώτη να περιέχει τα υπόλοιπα πεδία της εντολής.

Χωρίς να εισάγετε νέες υπομονάδες στη ΜΕΔ, εξηγήστε εάν και πώς μπορείτε να υποστηρίξετε την παραπάνω τροποποίηση, τόσο στην περίπτωση απλού, όσο και στην περίπτωση πολλαπλών κύκλων μηχανής για κάθε κύκλο εντολής, για καθεμία από τις εντολές addi, lw, sw και beq. Πιο συγκεκριμένα, τι προσθήκες χρειάζεστε στη ΜΕΔ για την υποστήριξη των εντολών αυτών; Χρειάζεστε κάποιους νέους δρόμους μεταφοράς πληροφορίας; Χρειάζεστε κάποια νέα σήματα ελέγχου;
Στην περίπτωση των πολλαπλών κύκλων μηχανής, να δώσετε την απαραίτητη προσθήκη στη μηχανή καταστάσεων που να περιγράφει την εκτέλεση των εντολών.

Άσκηση 4

Θεωρήστε τη ΜΕΔ της αρχιτεκτονικής MIPS καλωδιωμένης λογικής για κύκλο εντολής πολλαπλών κύκλων μηχανής. Στην απλουστευμένη υλοποίηση που μελετήσαμε στο μάθημα, δεν υπήρχαν εντολής πολλαπλασιασμού και διαίρεσης σταθερής υποδιαστολής. Οι εντολές αυτές παραδοσιακά στην αρχιτεκτονική MIPS έχουν ως ορίσματα δύο κατ' ευθείαν τελούμενα εισόδου, ενώ ως τελούμενο εξόδου έχουν τον υπονοούμενο διπλό καταχωρητή ειδικού σκοπού *hi/lo*. Για τον πολλαπλασιασμό, έξοδος είναι το διπλού εύρους αποτέλεσμα της πράξης, ενώ για τη διαίρεση, έξοδοι είναι το πηλίκο (στον *lo*) και το υπόλοιπο (στον *hi*). Οι τιμές που περιέχονται σε κάθε τμήμα του καταχωρητή *hi/lo* λαμβάνονται με τη βοήθεια των εντολών mfhi και mflo, οι οποίες ως μοναδικό όρισμα δέχονται τον καταχωρητή όπου αντιγράφεται το περιεχόμενο του *hi* ή του *lo*, αντίστοιχα. Αν θέλουμε να υποστηρίξουμε τέτοιες εντολές στη

ΜΕΔ της αρχιτεκτονικής MIPS, είναι απαραίτητο να προσθέσουμε στη ΜΕΔ μια μονάδα πολλαπλασιασμού/διαίρεσης, με ενσωματωμένο σε αυτή το διπλό καταχωρητή *hi/lo*. Θεωρήστε ότι η νέα μονάδα με την εντολή mult υλοποιεί πολλαπλασιασμό με εκτέλεση διάρκειας 4 κύκλων μηχανής, ενώ με την εντολή div υλοποιεί διαίρεση με εκτέλεση διάρκειας 33 κύκλων μηχανής. Στον πολλαπλασιασμό ο *lo* λαμβάνει τιμή στο τέλος του 3ου κύκλου, ενώ στη διαίρεση ο *lo* λαμβάνει τιμή στο τέλος του 32ου κύκλου.

Α. Δώστε το σχηματικό διάγραμμα της ΜΕΔ που προκύπτει με την ενσωμάτωση της παραπάνω μονάδας πολλαπλασιασμού/διαίρεσης, όπου να φαίνεται πώς η νέα μονάδα συνδέεται στην υπάρχουσα ΜΕΔ. Εξηγήστε όποιες τροποποιήσεις απαιτούνται στην υπάρχουσα ΜΕΔ. Δώστε έναν πίνακα με όλα τα σήματα ελέγχου – υπάρχοντα, τροποποιημένα υπάρχοντα ή καινούργια – που χρησιμοποιούνται για την υποστήριξη των τεσσάρων νέων εντολών και δείξτε τις προσθήκες στη συνολική μηχανή καταστάσεων της αρχιτεκτονικής για τις εντολές αυτές. Ελαχιστοποιήστε τη συνολική διάρκεια των εντολών. Ειδικότερα, οι mult και div πρέπει να εισέρχονται στους κύκλους εκτέλεσης της αντίστοιχης πράξης, και οι mfhi και mflo πρέπει να εισέρχονται στη φάση αποθήκευσης αποτελέσματος, αμέσως μετά τη φάση αποκωδικοποίησης.

Β. Θεωρήστε τον ακόλουθο κώδικα C:

```
x = 0;
for (i = 0; i < n; i++)
    x += c[i] ? a[i] * b[i] / c[i] : a[i];
y = x % n;
```

όπου a, b, c είναι διανύσματα 32-bit ακεραίων, και i, x, y, n είναι βαθμωτές 32-bit ακέραιες μεταβλητές. Αν τα διανύσματα a, b και c βρίσκονται τοποθετημένα στη μνήμη με αρχικές διευθύνσεις που περιέχονται στους καταχωρητές $s0, $s1 και $s2, αντίστοιχα, να γράψετε τον αντίστοιχο κώδικα σε συμβολική γλώσσα MIPS, χρησιμοποιώντας *μόνο* πραγματικές εντολές και *όχι* ψευδοεντολές. Ποιος είναι ο ελάχιστος και ποιος είναι ο μέγιστος αριθμός εντολών MIPS που εκτελούνται από αυτόν τον κώδικα; Ποιος είναι ο μέσος αριθμός εντολών που εκτελούνται από τον κώδικα, εάν γνωρίζετε ότι ο πίνακας c περιέχει μηδενικές τιμές με πιθανότητα 10%; Για κάθε περίπτωση μέγιστου, ελάχιστου και μέσου αριθμού εντολών, πόσοι κύκλοι μηχανής απαιτούνται για την εκτέλεση του κώδικα, σε τι ποσοστό αυτοί οι κύκλοι καταναλώνονται σε πολλαπλασιασμό ή διαίρεση, ποιο είναι το μέσο CPI που προκύπτει, και ποιος είναι ο χρόνος εκτέλεσης, αν αυτή γίνεται σε μηχανή 2GHz;

Γ. Θεωρήστε τώρα μια νέα εντολή

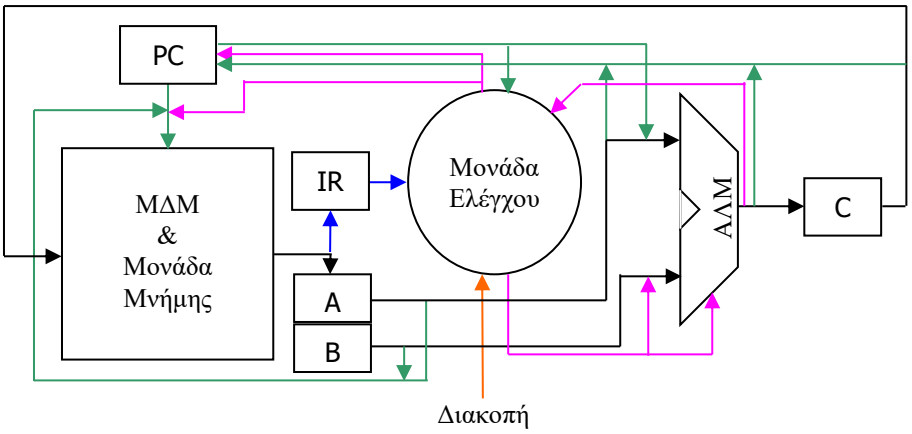
```
muldiv $rs,$rt,$rd
```

η οποία συνδυάζει τις δύο πράξεις πολλαπλασιασμού και διαίρεσης, έτσι ώστε η διαίρεση να χρησιμοποιεί ως διαιρετέο το διπλού εύρους αποτέλεσμα του πολλαπλασιασμού για επίτευξη μεγαλύτερης ακρίβειας, σε βάρος όμως του χρόνου εκτέλεσης της διαίρεσης, που τώρα απαιτεί 64 κύκλους. Ο πολλαπλασιασμός γίνεται μεταξύ των καταχωρητών $rs και $rt, ενώ ο διαιρέτης της διαίρεσης λαμβάνεται από τον καταχωρητή $rd που για να διαβαστεί απαιτείται τροποποίηση στην 1η είσοδο ανάγνωσης του ΦΚ, ώστε μέσω κατάλληλου πολυπλέκτη να επιλέγεται το πεδίο rd της λέξης εντολής αντί του πεδίου rs. Να τροποποιήσετε κατάλληλα το διάγραμμα της ΜΕΔ του ερωτήματος Α, να συμπληρώσετε τον πίνακα σημάτων ελέγχου με όποια σήματα απαιτούνται, και να δείξτε τις νέες καταστάσεις της μηχανής καταστάσεων, για υποστήριξη της νέας εντολής. Στη συνέχεια να επαναλάβετε το ερώτημα Β, χρησιμοποιώντας όμως τη νέα εντολή muldiv αντί των κλασικών mult και div, όπου αυτό έχει νόημα.

Άσκηση 5

Στην άσκηση αυτή θα μελετήσουμε μια ΜΕΔ αρχιτεκτονικής μνήμης-μνήμης με έλεγχο καλωδιωμένης λογικής πολλαπλών κύκλων μηχανής ανά κύκλο εντολής. Ένα σχηματικό διάγραμμα της ΜΕΔ αυτής δίνεται παρακάτω, όπου με μαύρο σημειώνονται οι δρόμοι γενικών

δεδομένων, με μπλε οι δρόμοι εντολών, με πράσινο οι δρόμοι διευθύνσεων, με μωβ οι δρόμοι ελέγχου (συμπεριλαμβανομένων των άμεσων τελούμενων και κατ' ευθείαν διευθύνσεων) και με πορτοκαλί ο δρόμος εξωτερικών διακοπών.



Η ΜΕΔ δεν περιέχει κανέναν καταχωρητή γενικού σκοπού και οι λειτουργίες των εντολών της ξεκινούν και τελειώνουν στη μνήμη. Περιέχει όμως 5 καταχωρητές ειδικού σκοπού: το μετρητή προγράμματος PC, τον καταχωρητή IR για την αποθήκευση της τρέχουσας εντολής, τον καταχωρητή C για την προσωρινή αποθήκευση της εξόδου της ΑΛΜ πριν αυτή σταλεί στη μνήμη, και τους καταχωρητές A και B για την προσωρινή αποθήκευση δεδομένων πριν αυτά σταλούν στην ΑΛΜ. Η ΜΕΔ υποστηρίζει άμεση, κατ' ευθείαν, έμμεση, διπλά έμμεση, υπονοούμενη, καθώς και σχετική διευθυνσιοδότηση στις εντολές της.
Πέρα από τις εντολές της πιο πάνω ΜΕΔ που θα περιγράψουμε στη συνέχεια, η λογική λειτουργίας της ΜΕΔ είναι παρόμοια τόσο με τη ΜΕΔ MIPS που μελετήσαμε στο μάθημα, όσο και με τη ΜΕΔ συσσωρευτή που αναπτύσσεται στις λυμένες ασκήσεις.

Για να επεκτείνουμε την πιο πάνω ΜΕΔ για υποστήριξη πιο σύνθετης διευθυνσιοδότησης, εισάγουμε δύο καταχωρητές-δείκτες X και Y. Οι X και Y χρησιμοποιούνται για τις δεικτοδοτούμενες αναφορές στη μνήμη. Η δεικτοδοτούμενη διευθυνσιοδότηση συνδυάζεται με έμμεση, με διάφορους τρόπους, όπως φαίνεται στον πίνακα που ακολουθεί, όπου δίνονται αναλυτικά οι εντολές της αρχιτεκτονικής με τις αποδεκτές διευθυνσιοδοτήσεις τους:

Εντολή (τελούμενο)	Διευθυνσιοδότηση												
	Κατ' ευθείαν	Άμεση	Έμμεση	Διπλά έμμεση	Δεικτ. X	Δεικτ. Y	Δεικτ. X / έμμεση	Έμμεση / δεικτ. Y	Δεικτ. Y / έμμεση	Δεικτ. X / διπλά έμμεση	Έμμεση / δεικτ. Y	Υπονοούμενη	Σχετική
ALU(dest)	1	0	1	0	1	1	1	1	0	0	0	0	0
ALU(src1)	1	0	1	1	1	0	1	0	1	1	0	0	0
ALU(src2)	1	1	1	1	0	1	0	1	0	0	1	0	0
CMP(src1)	όπως ALU(src1)												
CMP(src2)	όπως ALU(src2)												
BRANCH	0	0	0	0	0	0	0	0	0	0	0	0	1
LDX	1	1	1	0	0	1	0	1	0	0	0	0	0
LDY	1	1	1	0	1	0	1	0	0	0	0	0	0
STX	1	0	1	0	0	1	0	1	0	0	0	0	0
STY	1	0	1	0	1	0	1	0	0	0	0	0	0
INCX	0	0	0	0	0	0	0	0	0	0	0	1	0
INCY	0	0	0	0	0	0	0	0	0	0	0	1	0
DECX	0	0	0	0	0	0	0	0	0	0	0	1	0

DECY	0	0	0	0	0	0	0	0	0	0	0	1	0
CMPX	όπως LDX												
CMPY	όπως LDY												
JMP	1	0	1	1	0	0	1	0	0	0	0	0	0
JAL	1	0	1	1	0	0	1	0	0	0	0	0	0
JMPY	0	0	0	0	0	0	0	0	0	0	0	1	0

όπου στους συνδυασμούς διευθυνσιοδοτήσεων η επίλυση αναφοράς ακολουθεί την αναγραφόμενη σειρά. Προσέξτε ότι ALU και BRANCH είναι ομάδες εντολών που αναπτύσσονται στη συνέχεια. Επίσης, στις εντολές ALU και CMP δίνονται ξεχωριστές διευθυνσιοδοτήσεις για καθένα από τα τελούμενά τους (εισόδου: src1 και src2, εξόδου: dest).

Από τις εντολές που αναγράφονται, οι εντολές της ομάδας ALU είναι οι ADD, SUB, OR και AND, οι οποίες εκτελούν πρόσθεση, αφαίρεση, λογικό άθροισμα και λογικό γινόμενο αντίστοιχα. Και οι τέσσερεις αυτές εντολές έχουν δύο τελούμενα εισόδου και ένα εξόδου, που όλα αναφέρονται στη μνήμη. Κατ' εξαίρεση, το δεύτερο τελούμενο εισόδου μπορεί να είναι άμεσο τελούμενο. Οι εντολές ALU εκτελούνται στην ΑΛΜ, και με την παραγωγή του αποτελέσματος παράγονται και οι τιμές 3 ψηφίων κατάστασης που αποθηκεύονται σε κατάλληλο καταχωρητή της μονάδας ελέγχου: το ψηφίο Z παίρνει τιμή 1 όταν παράγεται μηδενικό αποτέλεσμα, το ψηφίο C παίρνει τιμή 1 όταν παράγεται κρατούμενο εξόδου, και το ψηφίο N παίρνει τιμή 1 όταν παράγεται αρνητικό αποτέλεσμα. Σε κάθε άλλη περίπτωση τα ψηφία αυτά παίρνουν τιμή 0.

Η εντολή CMP εκτελεί σύγκριση για ισότητα δύο τιμών από τη μνήμη, ή μιας τιμής από τη μνήμη και ενός άμεσου τελούμενου. Το λογικό αποτέλεσμα καταχωρείται σε ένα τέταρτο ψηφίο κατάστασης της μονάδας ελέγχου, το ψηφίο Q.

Οι εντολές της ομάδας BRANCH είναι εντολές άλματος με συνθήκη, και είναι οι BRZ (Z=1), BRNZ (Z=0), BRN (N=1), BRNN (N=0), BRC (C=1), BRNC (C=0), BEQ (Q=1), BNE (Q=0), BXZ (X=0), BXNZ (X≠0), BYZ (Y=0) και BYNZ (Y≠0), όπου στις παρενθέσεις δίνονται οι αντίστοιχες συνθήκες άλματος. Παρατηρήστε ότι οι συνθήκες σχετίζονται με τα ψηφία κατάστασης ή τους καταχωρητές-δείκτες. Η διευθυνσιοδότηση όλων αυτών των εντολών είναι σχετική με την αυξημένη τιμή του PC.

Οι εντολές LDX και LDY φορτώνουν τιμές στους καταχωρητές X και Y από τη μνήμη ή από άμεσο τελούμενο, ενώ οι εντολές STX και STY αποθηκεύουν το περιεχόμενο του αντίστοιχου καταχωρητή στη μνήμη.

Οι εντολές INCX και INCY αυξάνει τον αντίστοιχο καταχωρητή κατά 1, ενώ οι DECX και DECY μειώνουν τον αντίστοιχο καταχωρητή κατά 1. Οι εντολές αυτές εκτελούνται με ομώνυμα σήματα ελέγχου σε πρόσθετους αθροιστές που είναι ενσωματωμένοι στους καταχωρητές-δείκτες, και όχι στην ΑΛΜ.

Οι εντολές CMPX και CMPY συγκρίνουν το περιεχόμενο του αντίστοιχου καταχωρητή για ισότητα με κάποια τιμή από τη μνήμη ή από άμεσο τελούμενο, αποθηκεύοντας το λογικό αποτέλεσμα της σύγκρισης στο ψηφίο Q. Οι εντολές αυτές, όπως και η παραπάνω CMP εκτελούνται στην ΑΛΜ με τη μικρολειτουργία "compare".

Η εντολή JMP εκτελεί άμεσο άλμα, ενώ η εντολή JAL εκτελεί άμεσο άλμα με σύνδεση μέσω του καταχωρητή Y. Με την ολοκλήρωση μια εντολής JAL, ο Y θα περιέχει την προηγούμενη αυξημένη τιμή του PC.

Τέλος η εντολή JMPY εκτελεί έμμεσο άλμα μέσω του Y.

Α. Τροποποιήστε το αρχικό διάγραμμα της ΜΕΔ με την προσθήκη των καταχωρητών X και Y, καθώς και την προσθήκη των αναγκαίων δρόμων πληροφορίας για την υποστήριξη όλων των διευθυνσιοδοτήσεων που περιλαμβάνουν δεικτοδότηση. Θεωρήστε ότι η διάρκεια κύκλου μηχανής ορίζεται από το μέγιστο μιας προσπέλασης μνήμης, ή μιας πράξης ΑΛΜ.

Β. Χρησιμοποιώντας σαν παράδειγμα τη ΜΕΔ συσσωρευτή που περιγράφεται στις λυμένες ασκήσεις, ορίστε τα σήματα ελέγχου της ΜΕΔ που σχεδιάσατε, και κατασκευάστε τους πίνακες μικρολειτουργιών και σημάτων ελέγχου για όλες τις εντολές της αρχιτεκτονικής. Κάθε εντολή και διευθυνσιοδότηση να ολοκληρώνεται στον ελάχιστο αριθμό φάσεων. Προσπαθήστε να βάλετε στη φάση αποκωδικοποίησης μικρολειτουργίες που μπορούν να γίνουν προκαταβολικά, χωρίς να επηρεάζουν εντολές που δεν τις χρησιμοποιούν.

Γ. Σχεδιάστε τη μηχανή καταστάσεων της νέας ΜΕΔ. Επαναχρησιμοποιήστε όσες καταστάσεις μπορείτε. Βρείτε τις συνθήκες μετάβασης καταστάσεων για τη μηχανή που σχεδιάσατε, και τις λογικές εκφράσεις για τα σήματα ελέγχου που ορίσατε. Προσέξτε ότι η λογική έκφραση του σήματος εγγραφής του PC περιλαμβάνει και τις συνθήκες των διακλαδώσεων.

Άσκηση 6

Σε πολλές εφαρμογές παράλληλης επεξεργασίας απαιτείται μια κατηγορία λειτουργιών υλοποιημένων σαν ακολουθίες μικρολειτουργιών που εκτελούνται σαν ένα αδιάσπαστο σύνολο (indivisible ή atomic operations). Αυτό σημαίνει ότι κατά την εκτέλεση αυτών των λειτουργιών δεν επιτρέπεται ανάμεσα στις μικρολειτουργίες της ακολουθίας να παρεμβάλλονται ξένες μικρολειτουργίες. Ένας τρόπος υλοποίησης τέτοιων αδιάσπαστων λειτουργιών είναι μέσω εντολών μηχανής της αρχιτεκτονικής, έτσι ώστε μία και μόνη εντολή να εκτελεί πλήρως μια τέτοια ακολουθία.
Θεωρήστε τη ΜΕΔ της απλοποιημένης αρχιτεκτονικής MIPS πολλαπλών κύκλων μηχανής ανά κύκλο εντολής, με έλεγχο τόσο καλωδιωμένης όσο και μικροπρογραμματισμένης λογικής, και υλοποιήστε σε αυτή – για κάθε περίπτωση ελέγχου – την εντολή:

```
add&c&swapl $rd,$ru,($rt),($rs)
```

η οποία δεδομένων δύο θέσεων μνήμης και ενός αριθμού, εκτελεί τη λειτουργία που περιγράφεται από τα ακόλουθα βήματα:

1. Διαβάζει τη θέση μνήμης στην οποία αναφέρεται με έμμεση διευθυνσιοδότηση χωρίς μετατόπιση μέσω του καταχωρητή $rs.
2. Προσθέτει την τιμή που διάβασε από τη μνήμη με την τιμή του $ru.
3. Διαβάζει τη θέση μνήμης στην οποία αναφέρεται με έμμεση διευθυνσιοδότηση χωρίς μετατόπιση μέσω του καταχωρητή $rt.
4. Συγκρίνει το αποτέλεσμα της πρόσθεσης με τη δεύτερη τιμή που διάβασε από τη μνήμη.
5. Αν το αποτέλεσμα της πρόσθεσης είναι μικρότερο από τη δεύτερη τιμή:
   - Αποθηκεύει τη δεύτερη τιμή στη θέση της πρώτης τιμής που διάβασε, δηλαδή στη διεύθυνση που υποδεικνύει ο καταχωρητής $rs.
   - Αποθηκεύει το αποτέλεσμα της πρόσθεσης στη θέση της δεύτερης τιμής που διάβασε, δηλαδή στη διεύθυνση που υποδεικνύει ο καταχωρητής $rt.
6. Διαφορετικά αποθηκεύει το αποτέλεσμα της πρόσθεσης στη θέση της πρώτης τιμής που διάβασε, δηλαδή στη διεύθυνση που υποδεικνύει ο καταχωρητής $rs.
7. Αποθηκεύει στον $rd το λογικό αποτέλεσμα της σύγκρισης.

Για την υλοποίηση της εντολής add&c&swapl τροποποιήστε κατάλληλα τους δρόμους πληροφορίας και τις επιλογές εισόδων υπομονάδων της ΜΕΔ, διευρύνοντας ή προσθέτοντας πολυπλέκτες όπου χρειαστεί. Όμως, δε μπορείτε να προσθέσετε υπομονάδες ή καταχωρητές ειδικού σκοπού, μπορείτε όμως να προσθέσετε επίτρεψη εγγραφής σε υπάρχοντες καταχωρητές.
Επίσης, δε μπορείτε να αυξήσετε τις θύρες ανάγνωσης του ΦΚ. Γι' αυτό διατίθενται δύο μικρολειτουργίες ανάγνωσής του, η πρώτη διαβάζει τους $rs και $rt, και η δεύτερη διαβάζει τους $rs και $ru, επιλέγοντας τα κατάλληλα πεδία ψηφίων από τη λέξη εντολής. Η επιλογή ανάγνωσης του ΦΚ γίνεται με το σήμα "read", το οποίο όταν ενεργοποιείται επιλέγει τη δεύτερη ανάγνωση, διαφορετικά επιλέγεται η πρώτη.
Υποθέστε ότι η σύγκριση μεταξύ δύο τιμών για μικρότερο γίνεται στην ΑΛΜ με τη μικρολειτουργία "slt". Το λογικό αποτέλεσμα της σύγκρισης αποθηκεύεται στον C και αντικατοπτρί-

ζεται και στο ψηφίο Zero της ΑΛΜ που γίνεται διαθέσιμο με το τέλος της αντίστοιχης μικρολειτουργίας.

Για την υλοποίηση με έλεγχο καλωδιωμένης λογικής να δώσετε τις διαδοχικές καταστάσεις που ενεργοποιούν τα σήματα ελέγχου για την εντολή add&c&swapl, καθώς και τις συνθήκες μετάβασης για όλες τις μεταβάσεις μεταξύ των καταστάσεων. Οι λογικές εκφράσεις των συνθηκών μετάβασης μπορούν να περιέχουν και το ψηφίο Zero.
Για την υλοποίηση με έλεγχο μικροπρογραμματισμένης λογικής, δώστε το αντίστοιχο μικροπρόγραμμα. Κατ' αντιστοιχία με την καλωδιωμένη λογική, οι μικροεντολές μπορούν να περιέχουν άλμα με συνθήκη με βάση το ψηφίο Zero της ΑΛΜ. Αν χρειαστεί, προσθέστε και νέα πεδία μικροεντολής, και ορίστε κατάλληλες μικρολειτουργίες γι' αυτά.
Να ελαχιστοποιήστε το χρόνο κύκλου εντολής και για τις δύο υλοποιήσεις της εντολής, χωρίς ωστόσο να αυξήσετε το χρόνο κύκλου μηχανής.

Άσκηση 7

Έστω η υποθετική εντολή MIPS vsub&add η οποία αφαιρεί τα στοιχεία ενός διανύσματος από τα στοιχεία δεύτερου διανύσματος, και στη συνέχεια προσθέτει τις διαφορές στα στοιχεία ενός τρίτου διανύσματος όπου αποθηκεύεται και το αποτέλεσμα, παράγοντας επίσης και το πλήθος των ίσων στοιχείων των δύο πρώτων διανυσμάτων. Η εντολή έχει τη μορφή:

```
vsub&add $rd,$rs,$rt,$ru
```

όπου ο καταχωρητής $rs περιέχει την αρχική *διεύθυνση* του πρώτου διανύσματος, ο καταχωρητής $rt περιέχει την αρχική *διεύθυνση* του δεύτερου διανύσματος, ο καταχωρητής $ru περιέχει την αρχική *διεύθυνση* του τρίτου διανύσματος, ενώ ο καταχωρητής $rd περιέχει αρχικά το μέγεθος των διανυσμάτων, και στο τέλος θα πρέπει να περιέχει το ζητούμενο πλήθος των ίσων τιμών.

Α. Γράψτε το πρόγραμμα συμβολικής γλώσσας MIPS για την υλοποίηση (εξομοίωση της συμπεριφοράς) της παραπάνω υποθετικής εντολής. Ποιος είναι ο μέγιστος αριθμός εντολών που θα εκτελεστούν για την πράξη μεταξύ τριών διανυσμάτων Ν στοιχείων; Για έλεγχο καλωδιωμένης λογικής πολλαπλών κύκλων μηχανής, ποιος είναι ο αριθμός κύκλων που αντιστοιχεί στο μέγιστο αριθμό εντολών;

Β. Μελετήστε την υλοποίηση της εντολής vsub&add σε μικροπρογραμματισμένη ΜΕΔ MIPS, τροποποιώντας τη ΜΕΔ, ώστε να υποστηρίζει τη ροή πληροφορίας που χρειάζεστε, αλλά όσο μπορείτε λιγότερο και χωρίς να προσθέσετε νέες υπομονάδες. Μπορείτε όμως να προσθέσετε καταχωρητές ειδικού σκοπού, να προσθέσετε επίτρεψη εγγραφής σε υπάρχοντες καταχωρητές, καθώς και να διευρύνετε ή να προσθέσετε πολυπλέκτες. Υποθέστε ότι δε μπορείτε να αυξήσετε τις θύρες ανάγνωσης και εγγραφής του ΦΚ, όμως διατίθενται δύο μικρολειτουργίες ανάγνωσής του, η πρώτη διαβάζει τους $rs και $rt, και η δεύτερη διαβάζει τους $rd και $ru, επιλέγοντας τα κατάλληλα πεδία ψηφίων από τη λέξη εντολής. Η επιλογή ανάγνωσης του ΦΚ γίνεται με το σήμα "read", το οποίο όταν ενεργοποιείται επιλέγει τη δεύτερη ανάγνωση, διαφορετικά επιλέγεται η πρώτη. Εγγραφές σε πολλαπλούς καταχωρητές πρέπει να γίνονται σε διαφορετικούς κύκλους μηχανής, υποθέτοντας ότι μπορείτε να λάβετε καταχωρητή εγγραφής από όλα τα πεδία της λέξης εντολής, με κατάλληλο σήμα επιλογής. Στον ίδιο κύκλο μηχανής μπορεί να ενεργοποιείται μία μικρολειτουργία ανάγνωσης και μία μικρολειτουργία εγγραφής του ΦΚ, με την εγγραφή να γίνεται στο πρώτο μισό και την ανάγνωση στο δεύτερο μισό του κύκλου. Δώστε το τελικό διάγραμμα της ΜΕΔ.

Γ. Με βάση το πρόγραμμα του ερωτήματος Α και τις τροποποιήσεις της ΜΕΔ που προτείνατε στο ερώτημα Β, γράψτε ένα μικροπρόγραμμα για την υλοποίηση της εντολής vsub&add. Αν χρειαστεί, προσθέστε νέα πεδία μικροεντολής, και ορίστε το συμβολισμό των αντίστοιχων μικρολειτουργιών. Το πεδίο άλματος της μικροεντολής μπορεί να περιέχει άλμα με συνθήκη το ψηφίο Zero της ΑΛΜ, και με πεδίο προορισμού τη διεύθυνση προορισμού σε περίπτωση αληθούς συνθήκης. Προσπαθήστε να γράψετε όσο πιο σύντομο μικροκώδικα μπορείτε. Με-

τρήστε το μέγιστο αριθμό κύκλων που απαιτούνται για την πράξη μεταξύ τριών διανυσμάτων Ν στοιχείων, και σχολιάστε το αποτέλεσμα σε σχέση με εκείνο του πιο πάνω ερωτήματος Α.

Υπόδειξη: Θεωρήστε ότι οι καταχωρητές $rs, $rt, $ru και $rd δεν είναι απαραίτητο να διατηρήσουν τις αρχικές τιμές τους.

Άσκηση 8

Θεωρήστε έναν επεξεργαστή με έλεγχο πολλαπλών κύκλων μηχανής για κάθε κύκλο εντολής. Σε έναν τέτοιο επεξεργαστή, δύο σημαντικές παράμετροι καθορίζουν την απόδοσή του: η διάρκεια του κύκλου μηχανής, και ο μέσος αριθμός κύκλων μηχανής ανά εντολή (CPI). Κατά τη σχεδίαση ενός τέτοιου επεξεργαστή, άλλοι προτιμούν να υλοποιούν υψηλή συχνότητα – κι επομένως μικρή διάρκεια κύκλου μηχανής, αλλά με κόστος μια υψηλή τιμή της παραμέτρου CPI, κι άλλοι προτιμούν να υλοποιούν χαμηλή τιμή της τελευταίας, αλλά με κόστος τη μεγαλύτερη διάρκεια του κύκλου μηχανής.
Θα μελετήσουμε στη συνέχεια την επίδραση των δύο παραμέτρων στην απόδοση τριών αρχιτεκτονικών MIPS με βάση τρία προγράμματα αξιολόγησης της απόδοσης, κατά την εκτέλεση των οποίων εμφανίστηκαν εντολές MIPS με ποσοστό ανά τύπο εντολής όπως δείχνει ο ακόλουθος πίνακας:

	ALU-register	ALU-immediate	Shift	Mul	Div	Load	Store	Branch	Jump	FP Add/Sub	FP Mul	FP Div	FP Other
Πρόγραμμα Α	21%	16%	4%	3%	1%	27%	12%	14%	2%	-	-	-	-
Πρόγραμμα Β	30%	21%	2%	-	-	25%	10%	10%	2%	-	-	-	-
Πρόγραμμα Γ	18%	9%	2%	-	-	24%	9%	12%	1%	15%	7%	2%	1%

Αρχιτεκτονική Μ1: Η αρχιτεκτονική αυτή βασίζεται σε εκείνη που μελετήσαμε στο μάθημα, με εντολές ΑΛΜ με κατ' ευθείαν τελούμενα (ALU-register), εντολές φόρτωσης (Load), αποθήκευσης (Store), διακλάδωσης (Branch) και άλματος (Jump). Οι εντολές ολίσθησης (Shift) και ΑΛΜ με άμεσο τελούμενο (ALU-immediate) απαιτούν τον ίδιο αριθμό κύκλων με τις εντολές ΑΛΜ με κατ' ευθείαν τελούμενα. Επιπλέον αυτών των εντολών, υποστηρίζονται και εντολές πολλαπλασιασμού (Mul) και διαίρεσης (Div), οι πράξεις των οποίων απαιτούν 8 και 32 κύκλους μηχανής, αντίστοιχα, καθώς και εντολές κινητής υποδιαστολής: πρόσθεσης/αφαίρεσης (FP Add/Sub), πολλαπλασιασμού (FP Mul), διαίρεσης (FP Div) και άλλες (FP Other), οι πράξεις των οποίων απαιτούν 2, 4, 8 και 2 κύκλους μηχανής, αντίστοιχα. Η αρχιτεκτονική Μ1 έχει ρολόι με συχνότητα 4,1 GHz.

Αρχιτεκτονική Μ2: Παρόμοια με τη Μ1, με την τροποποίηση ότι οι εγγραφές στο ΦΚ συμβαίνουν στο τέλος του κύκλου μηχανής που παράγει το δεδομένο που γράφεται. Θεωρήστε ότι όλες οι εντολές πλην των τύπων Store, Branch και Jump κάνουν εγγραφή στο ΦΚ στον τελευταίο κύκλο μηχανής τους. Για παράδειγμα, σε εντολές τύπου Load, η εγγραφή στο ΦΚ θα γίνεται τώρα στο τέλος της φάσης προσπέλασης μνήμης, ενώ για εντολές τύπου ALU και Shift, η εγγραφή στο ΦΚ θα γίνεται στο τέλος της φάσης εκτέλεσης. Η αναμενόμενη αύξηση της διάρκειας του κύκλου μηχανής υποχρεώνει το ρολόι να έχει συχνότητα 3,3 GHz. Η μεγαλύτερη διάρκεια κύκλου μηχανής, όμως, επιτρέπει και την επανασχεδίαση των υπομονάδων των εντολών Mul και Div, έτσι ώστε να απαιτούν τώρα 4 και 12 κύκλους μηχανής για την εκτέλεση των πράξεών τους, αντίστοιχα.

Αρχιτεκτονική Μ3: Παρόμοια με τη Μ2, με την τροποποίηση ότι ο υπολογισμός της τελικής διεύθυνσης προσπέλασης για εντολές τύπου Load και Store γίνεται στον ίδιο κύκλο μηχανής με την προσπέλαση της μνήμης. Η διάρκεια του κύκλου μηχανής αυξάνεται κι άλλο, ώστε η πρόσθεση που απαιτείται για τον υπολογισμό της διεύθυνσης να μπορεί να χωρέσει στον ίδιο κύκλο με την προσπέλαση, οπότε το ρολόι έχει συχνότητα 2,1 GHz. Ως αποτέλεσμα της μεγαλύτερης διάρκειας κύκλου μηχανής, ο αριθμός κύκλων για όλες τις εντολές κινητής υποδιαστολής μειώνεται κατά έναν, καθώς η τελευταία φάση τους ενσωματώνεται στην προηγούμενη. Καμία αλλαγή δε συμβαίνει σε άλλες εντολές.

Α. Βρείτε τον αριθμό κύκλων μηχανής για κάθε τύπο εντολής σε καθεμιά από τις τρεις παραπάνω αρχιτεκτονικές. Στη συνέχεια, χρησιμοποιώντας τις συχνότητες εμφάνισης των εντολών στα συγκεκριμένα προγράμματα αξιολόγησης, υπολογίστε την παράμετρο CPI για κάθε αρχιτεκτονική και κάθε πρόγραμμα. Ποια αρχιτεκτονική φαίνεται να έχει την καλύτερη απόδοση για κάθε πρόγραμμα ξεχωριστά, με βάση το μέσο χρόνο ανά εντολή; Ποια αρχιτεκτονική θα προτιμούσατε αν τις εφαρμογές σας αντιπροσώπευε ένα φορτίο από μία εκτέλεση του προγράμματος Α, δύο εκτελέσεις του προγράμματος Β και πέντε εκτελέσεις του προγράμματος Γ, υποθέτοντας ότι και τα τρία προγράμματα εκτελούν τον ίδιο αριθμό εντολών; Τι συμπέρασμα βγάζετε σχετικά με την αξιολόγηση μιας αρχιτεκτονικής; Ποιες άλλες παράμετροι πρέπει να λαμβάνονται υπόψη για μια αξιολόγηση;

Β. Η τροποποίηση της αρχιτεκτονικής Μ3 είναι ιδιαίτερα αρεστή στους σχεδιαστές του επεξεργαστή, όμως η μεγάλη αύξηση της διάρκειας κύκλου μηχανής την καθιστά μη εφαρμόσιμη. Αντί αυτής, επιλέχθηκε η εισαγωγή πρόσθετων εντολών προσπέλασης μνήμης με διευθυνσιοδότηση χωρίς μετατόπιση. Έτσι, αντί η πρόσθεση να ενσωματώνεται στον κύκλο προσπέλασης, καταργείται όταν δε χρειάζεται. Η νέα αυτή διευθυνσιοδότηση αυξάνει τον κύκλο μηχανής λόγω πιο πολύπλοκης αποκωδικοποίησης, αλλά πολύ λιγότερο από ό,τι στην Μ3, και οδηγεί σε ρολόι 2,9 GHz. Οι εντολές κινητής υποδιαστολής τώρα δεν επηρεάζονται σε σχέση με την αρχιτεκτονική Μ2. Αν η νέα αυτή διευθυνσιοδότηση χρησιμοποιείται στο ίδιο ποσοστό εντολών Load και εντολών Store, βρείτε ποιο πρέπει να είναι τουλάχιστον αυτό το ποσοστό σε καθένα από τα τρία προγράμματα ξεχωριστά, καθώς και στο φορτίο του προηγούμενου ερωτήματος, ώστε η απόδοση της νέας αρχιτεκτονικής – έστω Μ4 – να ξεπερνά την απόδοση της Μ2. Στην περίπτωση του φορτίου θεωρήστε ότι το ποσοστό είναι το ίδιο και στα τρία προγράμματα.

Προσοχή: Οι θεωρητικές ασκήσεις παραδίνονται μόνο χειρόγραφες, κάθε άσκηση σε ξεχωριστές σελίδες. Η παράδοση συνοδεύεται από υποχρεωτική σύντομη εξέταση.

Β. Τρίτη Σειρά Ασκήσεων (Εργαστηριακές Ασκήσεις – Μέρος 1ο)

παράδοση στο τέλος του εξαμήνου

Άσκηση 1

Θεωρήστε την αναπαράσταση IEEE 754 των 32 bits για αριθμούς κινητής υποδιαστολής απλής ακρίβειας, σε μια αρχιτεκτονική MIPS που *δε* διαθέτει υλικό κινητής υποδιαστολής. Τα ερωτήματα που ακολουθούν σας ζητούν να γράψετε κώδικα συμβολικής γλώσσας για την αρχιτεκτονική αυτή, στον οποίο όλες οι λειτουργίες κινητής υποδιαστολής θα πρέπει να υλοποιηθούν με εντολές ακεραίων. Ο κώδικας θα πρέπει να γραφτεί για τον προσομοιωτή SPIM ή τον προσομοιωτή MARS. Για απλούστευση, αγνοήστε μη κανονικοποιημένους αριθμούς.

Α. Να γράψετε ένα υποπρόγραμμα fread, το οποίο να διαβάζει έναν αριθμό κινητής υποδιαστολής απλής ακρίβειας, που δίνεται στο δεκαδικό σύστημα σε μορφή συμβολοσειράς, και να τον μετατρέπει στο παραπάνω πρότυπο. Αποδεκτές συμβολοσειρές είναι όσες αποτελούνται από σύμβολα ψηφίων (‘0’-‘9’) και περιέχουν υποχρεωτικά υποδιαστολή ‘,’ με τουλάχιστον ένα ψηφίο πριν και μετά την υποδιαστολή, ή περιέχουν τουλάχιστον ένα ψηφίο και εκθέτη που δίνεται με το χαρακτήρα ‘E’ ή ‘e’ ακολουθούμενο από προαιρετικό πρόσημο και τουλάχιστον ένα ψηφίο, ή περιέχουν και υποδιαστολή και εκθέτη. Προαιρετικός χαρακτήρας προσήμου προηγείται του ακέραιου μέρους του αριθμού. Οι συμβολοσειρές έχουν μέγιστο μήκος 40 χαρακτήρων και τερματίζονται με το χαρακτήρα αλλαγής γραμμής ‘\n’. Το υποπρόγραμμα θα πρέπει να ελέγχει τη συμβολοσειρά αφού τη διαβάσει, και αν αυτή δεν είναι αποδεκτή, ή αν ο αριθμός βρίσκεται εκτός εύρους αναπαράστασης, θα πρέπει να επιστρέφεται η ειδική τιμή NaN. Κατά τη μετατροπή, ο αριθμός στρογγυλοποιείται προς τον πλησιέστερο άρτιο. Το αποτέλεσμα της μετατροπής επιστρέφεται στον καταχωρητή $v0.

Β. Να γράψετε ένα υποπρόγραμμα fwrite, το οποίο να εκτυπώνει έναν αριθμό κινητής υποδιαστολής απλής ακρίβειας σε δεκαδικό σύστημα, όταν ο αριθμός δίνεται στο παραπάνω πρότυπο. Ο αριθμός δίνεται στον καταχωρητή $a0 και – για τιμή εντός εύρους αναπαράστασης – θα πρέπει να εκτυπώνεται ως συμβολοσειρά με πρόσημο, ένα ακέραιο ψηφίο, υποδιαστολή ‘,’, τουλάχιστον ένα κλασματικό ψηφίο και εκθέτη αποτελούμενο από το χαρακτήρα ‘E’, πρόσημο και τουλάχιστον ένα ψηφίο. Αν το κλασματικό μέρος είναι 0, η υποδιαστολή και τα κλασματικά ψηφία πρέπει να παραλείπονται. Παρόμοια, αν ο εκθέτης είναι 0, πρέπει να παραλείπεται. Κατά τη μετατροπή στο δεκαδικό σύστημα θα πρέπει να διατηρούνται όλα τα κλασματικά ψηφία του αριθμού. Για τις ειδικές τιμές NaN, $+\infty$ και $-\infty$ εκτυπώνεται αντί αριθμού κατάλληλο μήνυμα. Η εκτύπωση τερματίζεται με το χαρακτήρα ‘\n’.

Γ. Να γράψετε ένα υποπρόγραμμα vfread, το οποίο να διαβάζει ένα διάνυσμα n αριθμών κινητής υποδιαστολής απλής ακρίβειας, καλώντας n φορές το υποπρόγραμμα fread. Το υποπρόγραμμα διαβάζει πρώτα τον ακέραιο αριθμό n και μετά το διάνυσμα των n αριθμών κινητής υποδιαστολής. Το διάνυσμα κινητής υποδιαστολής αποθηκεύεται σε διαδοχικές θέσεις σε κάποιο χώρο μνήμης, με αρχική διεύθυνση και μέγεθος (σε bytes) που παρέχονται στο υποπρόγραμμα ως παράμετροι μέσω των καταχωρητών $a0 και $a1, αντίστοιχα. Για κάθε αριθμό τιμής NaN που επιστρέφει το υποπρόγραμμα fread, ο αριθμός n μειώνεται κατά 1, και το vfread προχωράει στον επόμενο αριθμό, χωρίς να κάνει αποθήκευση της τιμής. Το υποπρόγραμμα vfread πρέπει να είναι φιλικό προς το χρήστη, δηλαδή να τον κατευθύνει με κατάλληλα μηνύματα, καθώς και να εκτυπώνει κάθε αριθμό που διάβασε, καλώντας το υποπρόγραμμα fwrite. Η τελική τιμή του n επιστρέφεται στον καταχωρητή $v0.

Δ. Να γράψετε ένα υποπρόγραμμα vfwrite, το οποίο να εκτυπώνει ένα διάνυσμα n αριθμών κινητής υποδιαστολής απλής ακρίβειας, καλώντας n φορές το υποπρόγραμμα fwrite. Ο ακέραιος αριθμός n παρέχεται στο υποπρόγραμμα ως παράμετρος μέσω του καταχωρητή $a0. Το διάνυσμα κινητής υποδιαστολής βρίσκεται αποθηκευμένο σε διαδοχικές θέσεις σε κάποιο χώρο μνήμης, με αρχική διεύθυνση που παρέχεται μέσω του καταχωρητή $a1.

Ε. Να γράψετε ένα υποπρόγραμμα vfsort, το οποίο να ταξινομεί τα στοιχεία ενός διανύσματος n αριθμών κινητής υποδιαστολής απλής ακρίβειας που βρίσκονται στο παραπάνω πρότυ-

πο σε φθίνουσα σειρά *απόλυτου* μεγέθους. Ο ακέραιος αριθμός n παρέχεται στο υποπρόγραμμα ως παράμετρος μέσω του καταχωρητή $a0. Το διάνυσμα κινητής υποδιαστολής βρίσκεται αποθηκευμένο σε διαδοχικές θέσεις σε κάποιο χώρο μνήμης, με αρχική διεύθυνση που παρέχεται στο υποπρόγραμμα ως παράμετρος μέσω του καταχωρητή $a1. Το υποπρόγραμμα vfsort αποθηκεύει το ταξινομημένο διάνυσμα στον ίδιο χώρο μνήμης στον οποίο βρισκόταν το αρχικό διάνυσμα, και μετά καλεί το υποπρόγραμμα vfwrite για να το εκτυπώσει.

ΣΤ. Να γράψετε ένα υποπρόγραμμα vfsadd, το οποίο να προσθέτει μεταξύ τους τα στοιχεία ενός διανύσματος n αριθμών κινητής υποδιαστολής απλής ακρίβειας που βρίσκονται στο παραπάνω πρότυπο, και είναι ταξινομημένοι σε φθίνουσα σειρά απόλυτου μεγέθους. Ο ακέραιος αριθμός n παρέχεται στο υποπρόγραμμα ως παράμετρος μέσω του καταχωρητή $a0. Το διάνυσμα κινητής υποδιαστολής βρίσκεται αποθηκευμένο σε διαδοχικές θέσεις σε κάποιο χώρο μνήμης, με αρχική διεύθυνση που παρέχεται στο υποπρόγραμμα ως παράμετρος μέσω του καταχωρητή $a1. Η vfsadd θα πρέπει να εκμεταλλεύεται το γεγονός ότι οι n αριθμοί είναι ταξινομημένοι, ελαχιστοποιώντας τους υπολογισμούς που πρέπει να κάνει για την πρόσθεση. Το αποτέλεσμα της πρόσθεσης επιστρέφεται στον καταχωρητή $v0. Αν η πρόσθεση οδηγήσει σε υπερχείλιση, τότε η vfsadd επιστρέφει τιμή $+\infty$ ή $-\infty$, ενώ αν οδηγήσει σε υποχείλιση (ανεπάρκεια), τότε η vfsadd απλά επιστρέφει 0.

Ζ. Να γράψετε ένα πρόγραμμα, το οποίο να καλεί το υποπρόγραμμα vfread για να διαβάσει ένα διάνυσμα n αριθμών κινητής υποδιαστολής απλής ακρίβειας που βρίσκονται στο παραπάνω πρότυπο, στη συνέχεια να καλεί το υποπρόγραμμα vfsort για να ταξινομήσει τα στοιχεία του σε φθίνουσα σειρά απόλυτου μεγέθους, να καλεί το υποπρόγραμμα vfsadd για να υπολογίσει το άθροισμα των στοιχείων του, και τέλος να καλεί την fwrite για να εκτυπώσει το τελι-κό αποτέλεσμα. Στο τμήμα δεδομένων του προσομοιωτή, το πρόγραμμα πρέπει να δηλώνει με την οδηγία space το χώρο μνήμης στον οποίο τοποθετείται το διάνυσμα, και μπορεί να δηλώνει και οποιουσδήποτε βοηθητικούς χώρους θέλετε.

Άσκηση 2

Γράψτε σε συμβολική γλώσσα MIPS ένα πρόγραμμα προσομοίωσης για ένα υποσύνολο του συνόλου εντολών MIPS. Ειδικότερα, υλοποιήστε μια εικονική μηχανή MIPS με φάκελο καταχωρητών και μνήμη, η οποία να μπορεί να εκτελεί προγράμματα MIPS, τοποθετημένα στη μνήμη, από όπου θα τα ανακαλεί, θα τα αποκωδικοποιεί και θα τα εκτελεί, εντολή προς εντολή. Η υλοποίησή σας μπορεί να χρησιμοποιήσει οποιαδήποτε από τις εντολές MIPS, αλλά ο προσομοιωτής θα πρέπει να υποστηρίζει ένα υποσύνολο του συνόλου εντολών. Πιο συγκεκριμένα, θα πρέπει να υποστηρίζει τις εντολές:

`add, addu, sub, subu, slt, sltu, or, and, xor, addi, addiu, slti, sltiu, ori, andi, xori, lui, lw, sw, beq, bne, j, jr, jal, jalr, sll, srl, sra, sllv, srlv, srav`

Για την επαλήθευση του προσομοιωτή να χρησιμοποιήσετε ένα από τα δύο διαθέσιμα προγράμματα προσομοίωσης συμβολικής γλώσσας MIPS, το SPIM ή το MARS. Προσαρμόστε τον προσομοιωτή σας για το πρόγραμμα που προτιμάτε. Υλοποιήστε το φάκελο καταχωρητών και τη μνήμη της εικονικής μηχανής στο χώρο δεδομένων (data) του προγράμματος. Τοποθετήστε στη μνήμη της εικονικής μηχανής τουλάχιστον τρία προγράμματα MIPS. Τα προγράμματα θα πρέπει να είναι συμβολομεταφρασμένα με το χέρι και τοποθετημένα στη μνήμη σε μορφή γλώσσας μηχανής. Τα δεδομένα εισόδου των προγραμμάτων θα τοποθετηθούν επίσης στη μνήμη της μηχανής, μαζί με δήλωση χώρου (space) για πιθανά δεδομένα εξόδου.

Άσκηση 3

Θέλουμε να υλοποιήσουμε μια γρήγορη μονάδα πολλαπλασιασμού προσημασμένων αριθμών σταθερής υποδιαστολής. Ανάμεσα στις διάφορες τεχνικές που υπάρχουν θα θεωρήσουμε τις ακόλουθες:

1. Τεχνική μείωσης αριθμού μερικών αθροισμάτων με έλεγχο 3 bit του πολλαπλασιαστή για κάθε άθροισμα, χρησιμοποιώντας το βελτιωμένο αλγόριθμο Booth.
2. Τεχνική διατήρησης κρατουμένου για την πρόσθεση των μερικών γινομένων, με ελαχιστοποίηση του αριθμού των επιπέδων διατήρησης.
3. Τεχνική μερικής επικάλυψης, με ανεξαρτητοποίηση της διάταξης διατήρησης κρατουμένου από τον τελικό αθροιστή, έτσι ώστε να μπορούμε να ξεκινήσουμε ένα νέο πολλαπλασιασμό στη διάταξη διατήρησης κρατουμένου, μόλις μπούμε στον τελικό αθροιστή.
4. Τεχνική πρόσθεσης με πρόβλεψη κρατουμένου για τον τελικό αθροιστή. Η τεχνική αυτή υλοποιείται ιεραρχικά για μεγάλο αριθμό bit. Όμως, ανάλογα με το περιθώριο που έχουμε στο χρόνο καθυστέρησης για τον υπολογισμό του αθροίσματος, μπορούμε από ένα επίπεδο ιεραρχίας πρόβλεψης και πάνω να χρησιμοποιήσουμε αθροιστή διάδοσης κρατουμένου για απλούστευση της υλοποίησης. Αντί δηλαδή να χρησιμοποιήσουμε νέα διάταξη πρόβλεψης κρατουμένου για να συνδυάσουμε τα επιμέρους κρατούμενα σε ανώτερο επίπεδο πρόβλεψης, συνδέουμε τους αθροιστές σε απλή διάταξη διάδοσης κρατουμένου.

Υποθέστε ότι θέλουμε μια μονάδα πολλαπλασιασμού 64 × 64 bit, η οποία να υπολογίζει γινόμενο των 128 bit, χρησιμοποιώντας τις πιο πάνω τεχνικές. Υλοποιήστε τη μονάδα αυτή με τη βοήθεια μιας γλώσσας περιγραφής υλικού (VHDL ή Verilog), και του προγράμματος ModelSim. Προσέξτε ότι εφόσον η μονάδα είναι μη επαναληπτική, ο έλεγχος όλων των τριάδων ψηφίων γίνεται παράλληλα, και τα σήματα που παράγονται από τον έλεγχο οδηγούν τους αθροιστές διατήρησης κρατουμένου, καθορίζοντας δηλαδή αν σε κάθε επίπεδο θα γίνεται πρόσθεση ή αφαίρεση του πολλαπλασιαστέου ή του διπλάσιού του, ή αν δε θα γίνεται καμία πράξη. Η τελευταία επιλογή μπορεί να υλοποιηθεί απλά μηδενίζοντας την αντίστοιχη είσοδο του αθροιστή.

Μπορείτε να προχωρήσετε με βάση τα ακόλουθα βήματα:

1. Υπολογίστε πόσα *τουλάχιστον* επίπεδα πρόβλεψης χρειαζόμαστε στην υλοποίηση του τελικού αθροιστή, ώστε η καθυστέρηση αυτού του αθροιστή να είναι το πολύ ίση με την καθυστέρηση των επιπέδων διατήρησης κρατουμένου. Θεωρήστε ότι κάθε κύκλωμα πλήρους αθροιστή χαρακτηρίζεται από καθυστέρηση 3 πυλών, και ότι κάθε επίπεδο πρόβλεψης κρατουμένου συνδυάζει 4 κρατούμενα του αμέσως προηγούμενου επιπέδου για να υπολογίσει 1 σούπερ-κρατούμενο.
2. Υπολογίστε το πλήθος των εισόδων της υπομονάδας διατήρησης κρατουμένου, λαμβάνοντας υπόψη τόσο τις επιλογές του αλγόριθμου Booth, όσο και τη λέξη συμπληρωμάτων που απαιτείται για την υλοποίηση αφαίρεσης.
3. Σχεδιάστε στο χαρτί το συνολικό διάγραμμα της μονάδας. Προσέξτε την ελαχιστοποίηση των επιπέδων διατήρησης κρατουμένου, καθώς και τον αριθμό επιπέδων πρόβλεψης κρατουμένου που προκύπτει, σύμφωνα με τους υπολογισμούς καθυστέρησης που κάνατε.
4. Γράψτε το πρόγραμμα που αντιστοιχεί σε έναν επιμέρους αθροιστή διατήρησης κρατουμένου μεταβλητού εύρους, χρησιμοποιώντας στιγμιότυπα κυκλωμάτων πλήρων αθροιστών του 1 bit.
5. Ορίστε τα ενδιάμεσα σήματα που χρειάζονται για να συνδέσετε μεταξύ τους τους επιμέρους αθροιστές διατήρησης κρατουμένου και δημιουργήστε όλη τη μονάδα διατήρησης κρατουμένου με στιγμιότυπα επιμέρους αθροιστών, με τις κατάλληλες συνδέσεις κατά την αντιστοίχηση των παραμέτρων. Προσέξτε ιδιαίτερα τα περισσότερο σημαντικά ψηφία του κάθε επιπέδου, επειδή οι αριθμοί είναι προσημασμένοι.
6. Υλοποιήστε μέσω κατάλληλης διαδικασίας το κύκλωμα που παράγει τις εισόδους των επιμέρους υπομονάδων διατήρησης κρατουμένου, με βάση το βελτιωμένο αλγόριθμο Booth, συναρτήσει του πολλαπλασιαστέου και του πολλαπλασιαστή. Μην ξεχάσετε την παραγωγή της λέξης συμπληρωμάτων!
7. Υλοποιήστε τον αθροιστή πρόβλεψης κρατουμένου για την πρόσθεση του τελευταίου επιπέδου, προσθέτοντας έναν καταχωρητή στην είσοδό του για το μηχανισμό επικάλυψης, και συνδέοντάς τον με την υπόλοιπη μονάδα.
8. Ελέγξτε την ορθότητα του κυκλώματός σας με χρήση κατάλληλων διανυσμάτων δοκιμής, τα οποία να δοκιμάζουν τόσο μεμονωμένες, όσο και διαδοχικές επικαλυπτόμενες πράξεις.

Προσοχή: Οι εργαστηριακές ασκήσεις παραδίνονται μόνο ηλεκτρονικά. Η παράδοση συνοδεύεται από υποχρεωτική σύντομη επίδειξη.