Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Εαρινό Εξάμηνο 2023-2024

ΠΡΟΗΓΜΕΝΑ ΘΕΜΑΤΑ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Cheat Sheet

Ιωάννης (Χουάν) Τσαντήλας
03120883

Για καλύτερη κατανόηση, θα πρότεινα να έχετε παράλληλα ένα ανοιχτό λυμένο θέμα για να βρίσκετε αντίστοιχα παραδείγματα των περιπτώσεων που περιγράφω παρακάτω.

Tomasulo

Για να λύσουμε τον Tomasulo είναι καλό να κάνουμε το εξής πινακάκι με βάση τα δεδομένα της εκφώνησης (βάζω χρώματα για να δείξω ποιο αντιστοιχεί σε ποιο):

	RS	**FU**	**Cycles**
ADDD/SUBD	2	1	3
MULD/DIVD	2	1	5
BR/ADD/SUB/AND	4	4	2
MUL/DIV	1		2
LOAD	2		H:1, M: 5
STORE	2		H:1, M: 5

- Το σύστημα διαθέτει περιορισμένο αριθμό από reservation stations (RS). Συγκεκριμένα, περιέχει 2 RS για προσθέσεις/αφαιρέσεις και 2 RS για πολλαπλασιασμούς/διαιρέσεις floating point αριθμών. Αντίστοιχα, για integer αριθμούς περιλαμβάνονται 4 RS για εντολές διακλάδωσης, αριθμητικές και λογικές εντολές καθώς και 1 RS για πολλαπλασιασμούς/διαιρέσεις.
- Το σύστημα περιλαμβάνει 4 non-pipelined functional unit για πράξεις integer αριθμών. Όλες οι εντολές μεταξύ integer αριθμών διαρκούν 2 κύκλους.
- Το σύστημα περιλαμβάνει 2 non-pipelined floating point functional units, 1 για ADD/SUBD και 1 για MULD/DIVD. Οι εντολές πρόσθεσης/αφαίρεσης διαρκούν 3 κύκλους, ενώ οι εντολές πολλαπλασιασμού/διαίρεσης 5 κύκλους.
- Για τις εντολές αναφοράς στη μνήμη, στο στάδιο EX γίνεται τόσο ο υπολογισμός της διεύθυνσης αναφοράς όσο και η προσπέλαση στη μνήμη. Το σύστημα περιλαμβάνει ένα Load και ένα Store Queue καθένα από τα οποία διαθέτει 2 θέσεις. Οι εντολές χρησιμοποιούν ένα ξεχωριστό pipelined functional unit για τον υπολογισμό της διεύθυνσης και διαρκούν 1 κύκλο στην περίπτωση Hit στην cache και 5 κύκλους σε περίπτωση Miss.
- Ο **ROB έχει 9 θέσεις**.
- **BHR=0**. Δίνεται επίσης το FSM διάγραμμα του 2-bit predictor, ο οποίος προβλέπει **T για τιμές ≥ 2** και NT για τις υπόλοιπες.

	0 = 00	**1 = 01**	**2 = 10**	**3 = 11**
0	00	00	01	01
1	01	11	10	00

$$00 \underset{N}{\overset{T}{\rightleftarrows}} 01 \underset{N}{\overset{T}{\rightleftarrows}} 10 \underset{N}{\overset{T}{\rightleftarrows}} 11$$

- Το σύστημα περιλαμβάνει μια fully associative cache μεγέθους 32B με block size 16 bytes και πολιτική αντικατάστασης LRU. Αρχικά, η cache είναι άδεια.
- Οι καταχωρητές R1, R2 περιέχουν τις διευθύνσεις των $1^{ων}$ στοιχείων των πινάκων A και B αντίστοιχα, στους οποίους έχουν αποθηκευτεί αριθμοί διπλής ακρίβειας (μήκους 8B ο καθένας) – *δηλαδή έχουμε R1=A[0], R2=B[0])*.

Λύση

Μας δίνεται στην αρχή πως η cache είναι άδεια. Είναι 32B με μέγεθος μπλοκ 16B, άρα 32/16 = 2 μπλοκς. Κάθε στοιχείο των πινάκων Α και Β όμως έχει μέγεθος 8B, άρα κάθε μπλοκ θα έχει 2 στοιχεία. Επομένως, για αρχή, η cache έχει τη μορφή:

	8B	**8B**
0		
1		

Και η LRU=0, δηλαδή δείχνει στην 1η γραμμή.

Γενικά, για να συμπληρώσουμε τον πίνακα με τα IS, EX κτλπ, θα πρέπει κάθε φορά να ελέγχουμε τα εξής:

- **IS**: Έστω ότι είμαστε στην εντολή Χ. Έστω ότι το IS της προηγούμενης εντολής (της Χ-1 δηλαδή) είναι Ψ. Θεωρητικά, το IS της Χ θα είναι Ψ+1. Ωστόσο πρέπει να ελέγξουμε 2 πράγματα:
  - Ελέγχουμε εάν έχουμε χώρο στην ROB.
    - Για παράδειγμα, εάν η ROB έχει μέγεθος 9 και έχουμε 8 (ή και λιγότερες) εντολές πριν από την Χ, τότε μπαίνει.
    - Εάν έχουμε παραπάνω από 8 εντολές, τότε κοιτάμε το CMT των προηγούμενων 9 (δηλαδή των εντολών Χ-1 έως και Χ-9). Ξεκινάμε από την εντολή Χ-9 και τσεκάρουμε το CMT της. Αν CMT<X, τότε IS=X. Αν CMT>=X, τότε IS=CMT+1. Γράφουμε στα σχόλια Flush.
  - Κοιτάμε πόσα Reservation Stations έχουμε χρησιμοποιήσει μέχρι στιγμής σε όλο το μήκος του κώδικα.
    - Για παράδειγμα, για τις εντολές ADDD/SUBD έχουμε 2 RS. Έστω λοιπόν πως η Χ είναι ADDD. Επομένως, κοιτάμε πριν από αυτή, πόσες εντολές ADDD/SUBD έχουμε. Εάν έχουμε 0 ή 1, τότε δεν απαιτείται κανένας έλεγχος και μπαίνει κανονικά.
    - Εάν όμως έχουμε 2 ή παραπάνω, κοιτάμε το WR των 2 τελευταίων (έστω δηλαδή ότι έχουμε τις εντολές 3, 6, 8 και 10 – εμείς θα δούμε μόνο τις 8 και 10 - ενώ η δική μας είναι η 20). Αν το WR της παλαιότερης εκ των δύο (δηλαδή της 8) είναι WR<X, τότε IS=X. Αν WR>=X, τότε IS=WR+1.
  - Ανάμεσα στα τρία αυτά κριτήρια (δηλαδή απλά Ψ+1, ROB και RS) διαλέγουμε την μεγαλύτερη τιμή.
- **EX**: Έστω πως το IS της εντολής είναι Χ. Θεωρητικά, το EX της θα είναι Χ+1 έως Χ+1+Cycles-1. Ωστόσο:
  - Πρέπει να ελέγξουμε τα Functional Units.
    - Σε παρόμοιο πνεύμα με τα RS, κοιτάμε το είδος της εντολής και πόσα FUs έχουμε. Έστω πως η εντολή μας είναι ADDD και έχουμε 2 FUs. Κοιτάμε τις προηγούμενες εντολές του ίδιου είδους και ελέγχουμε πότε τελειώνουν τα EX τους (έστω δηλαδή ότι έχουμε τις εντολές 3, 6, 8 και 10 – εμείς θα δούμε μόνο τις 8 και 10 - ενώ η δική μας είναι η 20). Αν το τέλος του EX της παλαιότερης εκ των δύο (δηλαδή της εντολής 8 – έστω π.χ. πως είναι 3-5, κοιτάμε το 5) είναι EX<X, τότε το EX' (της εντολής μας) ξεκινά κανονικά στον κύκλο Χ+1. Διαφορετικά ξεκινά στον κύκλο EX+1.
  - Κοιτάμε τις εξαρτήσεις της εντολής. Μας νοιάζει μόνο η εξάρτηση RAW, δηλαδή κάποια προηγούμενη εντολή να γράφει πάνω σε μία μεταβλητή που χρησιμοποιούμε εδώ.
    - Για παράδειγμα, έστω η προηγούμενη εντολή LD F0, 0(R1) (η οποία γράφει πάνω στο F0) και η δική μας εντολή ADDD F4, F4. F0 (η οποία χρειάζεται το F0 για να κάνει το

άθροισμα). Πρόκειται δηλαδή για Read After Write hazard, αφού πρέπει να περιμένουμε να ολοκληρωθεί το γράψιμο στον F0 προτού τον διαβάσουμε.
    - Για αυτό, κοιτάμε το WR της εντολής (ή των εντολών, εάν έχουμε πολλές εξαρτήσεις) από την οποία εξαρτόμαστε. Εάν για το WR της προηγούμενης εντολής ισχύει WR<X+1, τότε το EX της δικής μας εντολής ξεκινά κανονικά στο X+1. Εάν για το WR της προηγούμενης εντολής ισχύει WR>=X+1, τότε το EX της δικής μας εντολής ξεκινά στο WR+1.
- **WR**: Έστω πως το EX της εντολής τελειώνει στον κύκλο X. Θεωρητικά, το WR μπορεί να ξεκινήσει στον X+1. Παρόλα αυτά, πρέπει να δούμε **όλη την στήλη** WR. Εάν υπάρχει κάποια εντολή με WR=X+1, τότε θα βάλουμε το αμέσως επόμενο (και στα σχόλια θα γράψουμε CDB Conflict).
    - Για παράδειγμα, έστω πως το EX της εντολής είναι 3-5. Άρα το WR μπορεί να είναι 6. Παρόλα αυτά, κοιτάζοντας την στήλη WR, βλέπουμε πως κάποια προηγούμενη εντολή έχει τιμή WR=6, και μία άλλη έχει τιμή WR=7, αλλά δεν υπάρχει η τιμή 8. Επομένως, θα βάλουμε WR=8.
- **CMT**: η τιμή του θα καθοριστεί από το WR της εντολής και το CMT της προηγούμενης εντολής, δηλαδή: $(CMT)_i = max((WR)_i, (CMT)_{i-1}) + 1$

Για να λύσουμε το θέμα του branch, πρέπει να βρούμε το prediction και το result. Το prediction δείχνεται παρακάτω, ενώ το result είναι αν πραγματικά παίρνουμε το branch ή όχι.

Για το prediction κοιτάμε την διεύθυνση της εντολής, π.χ. 0x1234567**C** BNEZ. Κρατάμε το τελευταίο ψηφίο της διεύθυνσης, εδώ είναι C και το μετατρέπουμε σε δυαδικό, εδώ είναι 1100. Κρατάμε το 3ο LSB, 1**1**00, και αυτό μας δείχνει την σειρά στον πίνακα:

	0 = 00	**1 = 01**	**2 = 10**	**3 = 11**
0	00	00	01	01
1	01	11	10	00

Το BHR μας δείχνει τη στήλη. Π.χ. έστω πως το BHR είναι 1, θα διαλέγαμε δηλαδή σαν prediction 11. Μας λέει η εκφώνηση πως για τιμές ≥2 δίνει T (taken), επομένως θα παίρναμε το branch. Εδώ έχουμε 2 περιπτώσεις:

- Το prediction και το result ταυτίζονται (επιτυχής πρόβλεψη): δεν κάνουμε καμία αλλαγή.
- Το prediction και το result δεν ταυτίζονται (αποτυχία): θα πρέπει να αλλάξουμε το BHR προς τα αριστερά κατά 1 (δηλαδή να γίνει 0, σε επόμενη διόρθωση θα γίνει 11 κ.ο.κ.), και το κουτάκι του πίνακα με βάση το σχεδιάγραμμα:

$$00 \underset{N}{\overset{T}{\rightleftarrows}} 01 \underset{N}{\overset{T}{\rightleftarrows}} 10 \underset{N}{\overset{T}{\rightleftarrows}} 11$$

Έστω πως είχαμε επιλέξει το 11, (2η στήλη, 2η γραμμή) δηλαδή το prediction ήταν Taken ενώ το result ήταν Not Taken. Με βάση το παραπάνω, το 11 πρέπει να γίνει 10:

	0 = 00	**1 = 01**	**2 = 10**	**3 = 11**
0	00	00	01	01
1	01	~~11~~ 10	10	00

Το branch θα ολοκληρωθεί σε κάποιον κύκλο του WR. Μέχρι αυτόν τον κύκλο, οι εντολές του branch θα εκτελεστούν. Έστω π.χ. πως το WR του branch γίνεται στον κύκλο 20. Οι επόμενες εντολές του branch θα αρχίζουν να εκτελούνται, αλλά όταν φτάσουμε σε κάποιο σημείο συμπλήρωσης του πίνακα να γράψουμε

τιμή μεγαλύτερη του 20, θα γράψουμε Χ. Αυτό θα συμβεί μέχρι το IS κάποιας εντολής να φτάσει 20, όπου εκεί θα βάλουμε σε όλα τα πεδία Χ και θα γράψουμε στα σχόλια Flush. Πλέον αυτές οι «καμένες» εντολές δεν μετρούν στο ROB και συνεχίζουμε στο μέρος του κώδικα όπου δεν θα παίρναμε το branch.