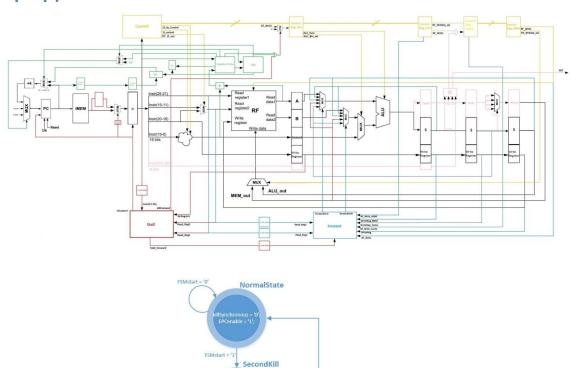
Αναφορά Εργαστηρίου 6

LAB31231454

Μανώλης Πετράκος	
Μιχαήλ Δακανάλης	

Προεργασία



Περιγραφή Άσκησης

Το πρώτο μέρος της άσκησης ήταν να δημιουργηθεί μία μονάδα που θα αναγνωρίζει και θα αντιμετωπίζει τα exceptions στέλνοντας το πρόγραμμα στον exception handler. Οι εξαιρέσεις που αντιμετωπίζονται είναι οι λανθασμένες εντολές και η πρόσβαση σε θέσεις μνήμης που δεν υπάρχουν.

ThirdKill

WaitForJump

Καταρχήν πρέπει στο decode να υπάρχει ο PC της εντολής που βρίσκεται στο συγκεκριμένο επίπεδο. Αποθηκεύουμε την τιμή στον καταχωρητή EPC ενώ την έχουμε καθυστερήσει κατάλληλα, ώστε να συμβαδίζει με την εντολή. Επίσης υπάρχει και ένα ασύγχρονο κύκλωμα που αναγνωρίζει τα exceptions κοιτώντας την εντολή και την διεύθυνση μνήμης που

παράγεται. Σε περίπτωση που εμφανιστεί ένα exception κάνει τέσσερις διαφορετικές λειτουργίες. Σκοτώνει την λανθασμένη εντολή, βάζει την κατάλληλη τιμή στον cause register, θέτει στον PC την διεύθυνση του κατάλληλου handler και ενεργοποιεί την FSM.

Στην κανονική λειτουργία του κυκλώματος η FSM έχει το enable των EPC και Cause Register ενεργό. Σε περίπτωση που εμφανιστεί ένα exception θα το απενεργοποιήσει ώστε να μείνουν οι τιμές της εντολής που προκάλεσε το πρόβλημα. Επίσης θα σκοτώσει τις επόμενες δύο εντολές που έχουν ήδη φορτωθεί στο pipeline. Τέλος θα μείνει σε μία κατάσταση όπου οι εντολές θα εκτελούνται κανονικά αλλά θα κρατάει τους EPC και Cause Register σταθερούς μέχρι να εμφανιστεί μία jump_epc.

Όταν έρθει μία jump_epc το ασύγχρονο κύκλωμα την αναγνωρίζει, θέτει στον PC την τιμή του EPC και γυρίζει την FSM στην αρχική της κατάσταση. Δημιουργήθηκε ένα πρόβλημα όπου το πρόγραμμα πήγαινε δύο φορές στον exception handler και μετά επέστρεφε στην κανονική λειτουργία. Για την αντιμετώπισή του, όταν η FSM βρίσκεται στην τελική κατάσταση, ο PC δεν παίρνει τις θέσεις των handler με την χρήση ενός πολυπλέκτη.

Το δεύτερο μέρος της άσκησης ζητάει να αντικατασταθεί η μνήμη δεδομένων με μία cache. Επίσης ο έλεγχος του hit/miss έπρεπε να γίνεται σε διαφορετικό κύκλο από την εύρεση των δεδομένων. Γι' αυτό προστέθηκε ένα παραπάνω επίπεδο στο Datapath και στο Control. Επίσης επεκτάθηκαν τα forward και τα stall ώστε να δουλεύουν και για τον επιπλέον κύκλο.

Η Cache δέχεται το index που έχει δημιουργηθεί από την εντολή και βγάζει το κατάλληλο block. Μετά υπάρχει ένα κύκλωμα το οποίο δέχεται το tag και το valid από την cache, το tag και το opcode της εντολής και παράγει το hit. Σε περίπτωση που η εντολή είναι load, αλλά το valid είναι 0 ή υπάρχει ασυμφωνία μεταξύ των tags υπάρχει miss και σκοτώνουμε την εντολή.

Στον επόμενο κύκλο από το Data της cache διαλέγεται η λέξη που υποδεικνύει το word offset και τέλος γράφεται στον Register File.

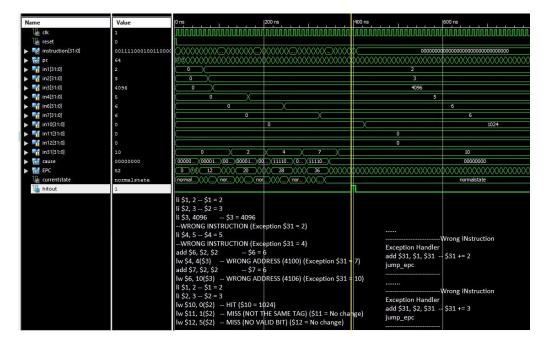
Η cache αποτελείται από τρία επιμέρους πεδία, το valid, το tag και το data. Το valid είναι 1 bit και δείχνει αν υπάρχουν έγκυρα δεδομένα στο block. Το tag αποτελείται από 3 bit και δείχνει από ποια θέση μνήμης προέρχονται τα δεδομένα. Το μέγεθός του εξαρτάται από την συνάρτηση: #address_bits $-\log_2(\text{cache}_{\text{size}}/\text{associativity})$. Το τελευταίο πεδίο είναι το data, έχει μέγεθος 128 bit και περιέχει 4 λέξεις.

Η διεύθυνση που παράγουν οι εντολές είναι 12 bytes και αποτελείται από 4 πεδία, το index τα byte και word offset και το tag. Το index αποτελείται από 5 bits ($log_2(size/associativity)$) και υποδεικνύει σε ποια θέση της cache θέλει να έχει πρόσβαση η εντολή. Το word offset αποτελείται από 2 bits και δείχνει ποια από τις 4 λέξεις του block θέλει η εντολή. Το byte offset δείχνει ποιο byte της λέξης χρειάζεται. Τέλος το tag αποτελείται από τα 3 bits που υπολείπονται και χρησιμοποιείται για τον έλεγχο ύπαρξης του σωστού block στην cache.

Κυμματομορφές

		175.000 ns													
Name	Value	120 ns	140 ns	160 ns	180 ns 200 ns	220 ns 240 ns 2									
lle cik	1														
Un reset	0														
- 📑 instruction[31:0]	000000000000000000000000000000000000000	000000 \(11100\)\(10101\)	(10000)(00111)(10	000 (10101	(0000000000000)(10000)(00111	X10000X00111X10000X10101X0000									
pc pc	24	20 24 28	256 X 260 X 3	264 268	24 (28) 32 (36	5 2 X 516 X 520 X 524 X 32									
in1[31:0]	2				2										
in2[31:0]	3				3										
in3(31:0)	4096				4096										
in4[31:0]	5	0		\rightarrow		-5									
in6[31:0]	0			(X									
in7[31:0]	0				0										
in10[31:0]	0				0										
in11[31:0]	0				0										
in12[31:0]	0				0										
in31[31:0]	2	0 \		2	X	4									
rause cause	00001111	00000000	000	01111	(00000000	11110000									
EPC	20	264 268 16		2 0	264 (268) 24	28									
le currentstate	normalstate	normalstate	secon (thirdkill)	waitforjump	normalstate	secon (thirdkill (waitforjump)									
la hitout	0														
		Ο ΕΡC δείχνει σε ποια Ο Cause Reg δείχνει τι Ο ΡC πηγαίνει στον ex Οι εντολές 24 και 28 έ; τις σκοτώνφυμε.	ον τύπο(IBUS) ception handler(25	66)	Ερχεται η jump_epc Ο PC γίνεται EPC + 4 Γίνονται 2 nop μέχρι να γυρίσει πίσω γιατί βρίσκονται μετά την jump_epc και έχουν μπει στο pipeline.	Ο Cause Reg δείχνει τον τύπο(AE Ο PC πηγαίνει στον exception handler(512)									

Name	Value	340 n		360	ns		380 n	s		400 n	s		420 r	ns	Si in	44	10 ns	ı Ör	1110	460 ns	a la	
Ve clk	1																					
Un reset	0		- 10		n					2									- 19/	- 60	100	
▶ 🎇 instruction[31:0]	00111100010011000	000000	0000)	11100	X11100)(001	11)	00111	001	11								00	00000	00000	000000	000000
▶ 📆 pc	64	40 (44	48	52	_X:	5	60	6	4 X	68	\supset	72	76	5)	80	X	84	X 8	B X	92	9
▶ 👫 in1[31:0]	2															2						
in2[31:0]	3															3						
▶ 📑 in3[31:0]	4096														4	096						
• 📑 in4[31:0]	5															5						
in6[31:0]	6															6						
• 📆 in7[31:0]	6															6						
• 📆 in10[31:0]	0						0						\Rightarrow									10
- 📆 in11[31:0]	0															0						
• 📆 in12[31:0]	0															0						
• 📆 in31[31:0]	10		7		X													10				
cause	00000000															00000	00					
▶ 📆 EPC	52	36 (520	524	X 40	(#)	48	5	2 X	56	\supset	60	64	1)	68	\supset	72	χ_{-2}	5 X	80	8
Currentstate	normalstate														norm	alstate	0					
1 hitout	1																					
									102 Στιο το h	4 μπ υπό nit είν	αίνε ολου ναι (load ι στο τες lo και δ	ν κατι ad επ δεν γρ	αχω _ι ειδή οάφε	ρητή είν	11. αι απ	οτυχ	χημέ				



Συμπεράσματα

Μάθαμε πως να διαχειριζόμαστε exceptions στο hardware και πως να δημιουργήσουμε αλλά και να προσθέσουμε στον επεξεργαστή μία μνήμη cache.