Register Number	Register Name	Purpose	
0	\$zero	Constant zero	
1	\$imm	Sign extended immediate	
2	\$v0	Result value	
3	\$a0	Argument register	
4	\$a1	Argument register	
5	\$t0	Temporary register	
6	\$t1	Temporary register	
7	\$t2	Temporary register	
8	\$t3	Temporary register	
9	\$s0	Saved register	
10	\$s1	Saved register	
11	\$s2 Saved register		
12	\$gp	Global pointer (static data)	
13	\$sp	Stack pointer	
14	\$fp	Frame Pointer	
15	\$ra	Return address	

שמות הרגיסטרים ותפקידם דומים למה שראינו בהרצאה ובתירגולים עבור מעבד MIPS, בהבדל אחד: רגיסטר מספר 1, imm\$, הינו רגיסטר מיוחד שלא ניתן לכתוב אליו, ותמיד מכיל את שדה \$\text{imm}\$, לאחר בצוע sign extension, כפי שקודד בהוראת האסמבלי. הוא מתעדכן עבור כל הוראה כחלק מפענוח ההוראה. רגיסטר 0 הינו זהותית אפס. הוראות אשר כותבות ל-\$\text{zero}\$ לא משנות את ערכו.

זיכרונות הוראות ונתונים

10 אינו ברוחב PC הינו ההוראות רגיסטר ה- PC לכן הינו ברוחב 1024 מיכרון ההוראות הינו ברוחב 20 סיביות ובעומק PC באחד במידה וההוראה מקודדת בשורה בודדת, או ביטים, והוראות עוקבות מקדמות את PC באתי שורות.

זיכרון הנתונים הינו ברוחב 32 ביטים ובעומק 4096 שורות. הכתובת לזיכרון הנתונים היא לכן ברוחב 12 ביטים. בניגוד למעבד MIPS, למעבד SIMP אין תמיכה ב- byte כל גישה לזיכרון הנתונים קוראת או כותבת מילה ברוחב 32 ביטים.

סט ההוראות וקידודם

: יש שני פורמטים לקידוד ההוראות SIMP למעבד

הפורמט הראשון הינו בשימוש עבור הוראות שאין בהן שימוש בשדה הקבוע. הוראות אלו מקודדות בשורה אחת בזיכרון ההוראות לפי חלוקת הביטים הבאה:

19:12	11:8	7:4	3:0
Opcode	rd	rs	rt

הוראות אשר משתמשות בקבוע מקודדות בשתי שורות בזיכרון ההוראות, כאשר השורה הראשונה זהה לפורמט הראשון, ובשורה השנייה מופיע הקבוע באופן הבא:

19:12	11:8	7:4	3:0
Opcode	rd	rs	rt
immediate (20 bits)			

האופקודים הנתמכים עייי המעבד ומשמעות כל הוראה נתונים בטבלה הבאה:

Opcode	Name	Meaning
Number		
0	add	R[rd] = R[rs] + R[rt]
1	sub	R[rd] = R[rs] - R[rt]
2	and	R[rd] = R[rs] & R[rt]
3	or	R[rd] = R[rs] R[rt]
4	xor	$R[rd] = R[rs] ^ R[rt]$
5	mul	R[rd] = R[rs] * R[rt]
6	sll	R[rd] = R[rs] << R[rt]
7	sra	R[rd] = R[rs] >> R[rt], arithmetic shift with sign extension
8	srl	R[rd] = R[rs] >> R[rt], logical shift
9	beq	if $(R[rs] == R[rt])$ pc = $R[rd]$ [low bits 9:0]
10	bne	if $(R[rs] != R[rt]) pc = R[rd] [low bits 9:0]$
11	blt	if $(R[rs] < R[rt])$ pc = R[rd] [low bits 9:0]
12	bgt	if $(R[rs] > R[rt])$ pc = R[rd] [low bits 9:0]
13	ble	if $(R[rs] \le R[rt])$ pc = R[rd] [low bits 9:0]

14	bge	if $(R[rs] \ge R[rt])$ pc = R[rd] [low bits 9:0]
15	jal	R[15] = next instruction address, pc = R[rd][9:0]
16	lw	R[rd] = DMEM[R[rs]+R[rt]]
17	sw	DMEM[R[rs]+R[rt]] = R[rd]
18	reti	PC = IORegister[7]
19	in	R[rd] = IORegister[R[rs] + R[rt]]
20	out	IORegister $[R[rs]+R[rt]] = R[rd]$
21	halt	Halt execution, exit simulator

קלט/פלט

המעבד תומך בקלט/פלט באמצעות הוראות in ו- out ו המעבד תומך בקלט/פלט באמצעות הוראות ב in המעבד תומך בקלט/פלט באמצעות הוראות ביציאה מריסט הם 0.

IORegister	Name	number	Meaning
Number		bits	
0	irq0enable	1	IRQ 0 enabled if set to 1, otherwise disabled.
1	irq1enable	1	IRQ 1 enabled if set to 1, otherwise disabled.
2	irq2enable	1	IRQ 2 enabled if set to 1, otherwise disabled.
3	irq0status	1	IRQ 0 status. Set to 1 when irq 0 is triggered.
4	irq1status	1	IRQ 1 status. Set to 1 when irq 1 is triggered.
5	irq2status	1	IRQ 2 status. Set to 1 when irq 2 is triggered.
6	irqhandler	10	PC of interrupt handler
7	irqreturn	10	PC of interrupt return address
8	clks	32	cyclic clock counter. Starts from 0 and
			increments every clock. After reaching
			0xffffffff, the counter rolls back to 0.
9	leds	32	Connected to 32 output pins driving 32 leds.
			Led number i is on when leds[i] == 1, otherwise
			its off.
10	reserved	32	Reserved for future use
11	timerenable	1	1: timer enabled
			0: timer disabled

12	timercurrent	32	current timer counter
13	timermax	32	max timer value
14	diskemd	2	0 = no command
			1 = read sector
			2 = write sector
15	disksector	7	sector number, starting from 0.
16	diskbuffer	12	Memory address of a buffer containing the
			sector being read or written. Each 512 byte
			sector will be read/written using DMA in 128
			words.
17	diskstatus	1	0 = free to receive new command
			1 = busy handling a read/write commad
18	monitoremd	1	0 = no command
			1 = write pixel to monitor
19	monitorx	9	X coordinate of pixel (0 – 351)
20	monitory	9	Y coordinate of pixel (0 – 287)
21	monitordata	8	Pixel luminance (gray) value (0 – 255)

פסיקות

מעבד SIMP תומך ב- 3 פסיקות: irq0, irq1, irq2. פסיקה 0 משויכת לטיימר, וקוד האסמבלי יכול לתכנת כל כמה זמן הפסיקה תתרחש.

פסיקה 1 משויכת לדיסק הקשיח המסומלץ, באמצעותה הדיסק מודיע למעבד כאשר סיים לבצע הוראת קריאה או כתיבה.

פסיקה 2 מחוברת לקו חיצוני למעבד irq2. קובץ קלט לסימולטור קובע מתי הפסיקה מתרחשת.

irq0status, irq1status, במחזור השעון בו הפסיקה מתקבלת, מדליקים את אחד הרגיסטרים וeq2status, במחזור השעון, ידלקו בהתאמה מספר וrq2status בהתאמה. אם מספר פסיקות מתקבלות באותו מחזור שעון, ידלקו בהתאמה מספר רגיסטרי סטטוס.

לפני בצוע כל הוראה, המעבד בודק את הסיגנל:

irq = (irq0enable & irq0status) | (irq1enable & irq1status) | (irq2enable & irq2status)

במידה ו- irq == 1, והמעבד לא נמצא כרגע בתוך שגרת הטיפול בפסיקה, המעבד קופץ לשגרת במידה ו- irq == 1, והמעבד לא נמצא כרגע בתוך ברגיסטר חומרה ביזיכרון נתונה ברגיסטר שעון זה PC = irqhandler במקום ב- PC = irqhandler במקורי באותו מחזור שעון ה- PC = irqhandler המקורי נשמר לתוך רגיסטר חומרה Irqreturn

שימו לב: במידה והמעבד נמצא כרגע באמצע בצוע הוראה שמשתמשת בקבוע ולוקחת שני מחזורי שעון, המעבד קודם כל יסיים את ההוראה ויבדוק את סיגנל ה- irq רק בסיום ההוראה.

לעומת זאת במידה ו- irq == 1 והמעבד עדיין נמצא בתוך שגרת הטיפול בפסיקה קודמת (כלומר irq == 1), המעבד יתעלם, לא יקפוץ וימשיך להריץ את הקוד כרגיל עדיין לא הריץ את הוראת ה-irq (כאשר המעבד יחזור מהפסיקה, הוא יבדוק שוב את irq ואם יהיה צורך יקפוץ שוב לשגרת הפסיקה).

קוד האסמבלי של שגרת הפסיקה יבדוק את הביטים של irqstatus, ולאחר טיפול מתאים בפסיקה יכבה את הביטים.

.PC = irqreturn שתציב, reti חזרה משגרת הפסיקה מתבצעת באמצעות הוראת

טיימר

מעבד SIMP תומר בטיימר של 32 ביטים, המחובר לפסיקה הוא מאופשר כאשר מעבד timerenable = 1 .

ערך מונה הטיימר הנוכחי שמור ברגיסטר חומרה timercurrent. בכל מחזור שעון שבו הטיימר מאופשר, רגיסטר timercurrent מקודם באחד.

במחזור השעון שבו timercurrent = timermax, מדליקים את irqstatus0. במחזור שעון זה .timercurrent מאפסים אותו חזרה לאפס.

דיסק קשיח

למעבד SIMP מחובר דיסק קשיח בגודל 64 קילובייט, המורכב מ- 128 סקטורים בגודל 512 בתים כל סקטור. הדיסק מחובר לפסיקה מספר 1, irq1, ומשתמש ב- DMA להעתקת הסקטור מהזיכרון לדיסק או להיפך.

תוכנו ההתחלתי של הדיסק הקשיח נתון בקובץ הקלט diskin.txt, ותוכן הדיסק בסיום הריצה ייכתב לקובץ diskout.txt.

לפני מתן הוראת קריאה או כתיבה של סקטור לדיסק הקשיח, קוד האסמבלי בודק שהדיסק פנוי לקבלת הוראה חדשה ע"י בדיקת רגיסטר חומרה diskstatus.

במידה והדיסק פנוי, כותבים לרגיסטר disksector את מספר הסקטור שרוצים לקרוא או לכתוב, ולרגיסטר diskbuffer את הכתובת בזיכרון. רק לאחר ששני רגיסטרים אלו מאותחלים, נותנים הוראת כתיבה או קריאה עייי כתיבה לרגיסטר חומרה diskemd.

זמן הטיפול של הדיסק בהוראת קריאה או כתיבה הוא 1024 מחזורי שעון. במהלך זמן זה יש להעתיק את תוכן הבפר לדיסק במידה והייתה כתיבה, או להיפך להעתיק את תוכן הסקטור לבפר אם הייתה קריאה.

כל עוד לא עברו 1024 מחזורי שעון מקבלת ההוראה, רגיסטר diskstatus יסמן שהדיסק עסוק.

לערך 0, והדיסק diskstatus ו- diskcmd לאחר 1024 לערך 0, והדיסק נוחדיסק מחזורי שעון, במקביל ישונו רגיסטר ה- irqstatus1 יודיע על פסיקה עייי הדלקת

מסך מחשב

למעבד SIMP מחובר מוניטור מונוכרומטי ברזולוציה 352x288 פיקסלים. כל פיקסל מיוצג עייי 8 ביטים שמייצגים את גוון האפור של הפיקסל (luminance) כאשר 0 מסמן צבע שחור, 255 צבע לבן, וכל מספר אחר בתחום מתאר גוון אפור בין שחור ללבן באופן לינארי.

במסך יש frame buffer פנימי בגודל 352x288 המכיל את ערכי הפיסקלים שכעת מוצגים על המסך. בתחילת העבודה כל הערכים מכילים אפס.

רגיסטר monitorx מכיל את קואורדינטת ה- X של הפיקסל שאותו המעבד רוצה לכתוב. X=351 הינו הפיקסל השמאלי, ו- X=351

רגיסטר monitory מכיל את קואורדינטת ה- Y של הפיקסל שאותו המעבד רוצה לכתוב. Y=287 הינו הפיקסל העליון, ו- Y=287 הפיקסל התחתון.

רגיסטר monitordata מכיל ערך הפיקסל שאותו המעבד רוצה לכתוב.

רגיסטר monitorcmd משמש עבור כתיבה של פיקסל. במחזור השעון שבו יש כתיבה מחסונ monitorcmd באמצעות הוראת out, מתבצע עדכון של הפיסקל שתוכנו ברגיסטר monitordata על המסך.

.0 באמצעות in באמצעות monitoremd בריאה מרגיסטר

סימולטור

הסימולטור מסמלץ את לולאת ה- fetch-decode-execute. בתחילת הריצה PC-0. בכל איטרצייה מביאים את ההוראה הבאה בכתובת ה- PC, מפענחים את ההוראה בהתאם לקידוד, איטרצייה מביאים את ההוראה הבאה בכתובת ה- sign extension מעדכנים את רגיסטר אחד עייי בצוע PC+2 לערך PC+1 במידה ולא היה שימוש בקבוע, או PC+2 במידה ולא היה שימוש בקבוע, או PC+2 במידה והיה שימוש בקבוע, אלא אם כן בצענו הוראת קפיצה שמעדכנת את ה- PC לערך אחר. PC במידה והיא מהסימולטור מתבצע כאשר מבצעים את הוראת ה- HALT.

13 אשר מקבל command line application ויקומפל לתוך C ויקומפל command line parameters לפי שורת ההרצה הבאה:

sim.exe imemin.txt dmemin.txt diskin.txt irq2in.txt dmemout.txt regout.txt trace.txt hwregtrace.txt cycles.txt leds.txt monitor.txt monitor.yuv diskout.txt הקובץ imemin.txt הינו קובץ קלט בפורמט טקסט אשר מכיל את תוכן זיכרון ההוראות בתחילת הריצה. כל שורה בקובץ מכילה תוכן שורה בזיכרון ההוראות, החל מכתובת אפס, בפורמט של 5 ספרות הקסאדצימליות. במידה ומספר השורות בקובץ קטן מ- 1024, ההנחה הינה ששאר הזיכרון מעל הכתובת האחרונה שאותחלה בקובץ, מאופס. ניתן להניח שקובץ הקלט תקין. הקובץ אשר הזיכרון מעל הכתובת האחרונה שובירמט טקסט אשר מכיל את תוכן זיכרון הנתונים בתחילת הריצה. כל שורה בקובץ מכילה תוכן שורה בזיכרון הנתונים, החל מכתובת אפס, בפורמט של 8 ספרות הקסאדצימליות. במידה ומספר השורות בקובץ קטן מ- 4096, ההנחה הינה ששאר הזיכרון מעל הכתובת האחרונה שאותחלה בקובץ, מאופס. ניתן להניח שקובץ הקלט תקין.

הקובץ diskin.txt הינו קובץ קלט, באותו פורמט כמו diskin.txt, הקובץ הדיסק הינו קובץ הדיסק החילת הריצה.

הקובץ irq2in.txt הינו קובץ קלט, המכיל את מספרי מחזורי השעון שבהם קו הפסיקה החיצוני irq2in.txt אלה ל- 1, כל מחזור שעון כזה בשורה נפרדת בסדר עולה. הקו כל פעם עולה ל- 1 למחזור שעון בודד ואז יורד חזרה לאפס (אלא אם כן מופיעה שורה נוספת בקובץ עבור מחזור השעון הבא).

ארבעת קבצי הקלט צריכים להיות קיימים אפילו אם בקוד שלכם אין בהם שימוש (לדוגמא גם עבור קוד אסמבלי שאינו משתמש בדיסק הקשיח יהיה קיים קובץ קלט diskin.txt, כאשר מותר גם להשאיר את תוכנו ריק).

הקובץ dmemin.txt הינו קובץ פלט, באותו פורמט כמו לmemin.txt, שמכיל את תוכן זיכרון הקובץ הריצה.

הקובץ regout.txt הקובץ פלט, שמכיל את תוכן הרגיסטרים R2-R15 בסיום הריצה (שימו לב שאין להדפיס את הקבועים R0 ו- R1). כל שורה תיכתב באותו פורמט כמו שורה ב- dmemin.txt, 8 ספרות הקסאדצימליות.

הינו קובץ פלט, המכיל שורת טקסט עבור כל הוראה שבוצעה עייי המעבד trace.txt הקובץ בפורמט הבא:

PC INST R0 R1 R2 R3 R4 R5 R6 R7 R8 R9 R10 R11 R12 R13 R14 R15

כל שדה מודפס בספרות הקסאדצימליות. ה- PC הינו ה- Program Counter של ההוראה ומודפס ב- 3 ספרות הקסאדצימליות, ה- INST הינו קידוד ההוראה כפי שנקרא מהזיכרון, ומודפס ב- 5 ספרות הקסאדצימליות. ואח״כ יש את תוכן הרגיסטרים לפני ביצוע ההוראה (כלומר את תוצאת הביצוע ניתן לראות רק ברגיסטרים של השורה הבאה), כאשר כל רגיסטר מודפס ב- 8 ספרות הקסאדצימליות.

בשדה R0 יש לכתוב 8 אפסים. בשדה R1 יש לכתוב את תוכן ה- Immediate בשדה R0 בשדה היו כולם sign extension ל- 32 סיביות. למשל אם כל 20 הביטים בקבוע היו כולם אחדים, אז נכתוב עבור R1 את הערך FFFFFFF.

הקובץ hwregtrace.txt הינו קובץ פלט, המכיל שורת טקסט עבור כל קריאה או כתיבה hwregtrace.txt הקובץ לרגיסטר חומרה (באמצעות הוראות in ו- out) בפורמט הבא:

CYCLE READ/WRITE NAME DATA

כאשר השדה CYCLE הוא מספר מחזור השעון בדצימאלי.

השדה הבא מכיל READ או WRITE בהתאם להאם קוראים או כותבים לרגיסטר החומרה.

השדה NAME מכיל את שם רגיסטר החומרה כפי שמופיע בטבלה.

השדה DATA מכיל את הערך שנכתב או נקרא ב- 8 ספרות הקסאדצימליות.

הקובץ cycles.txt הינו קובץ פלט, שמכיל שתי שורות: שורה ראשונה מכילה את זמן הריצה הקובץ הריצה שעון, ושורה שנייה מכילה את מספר הוראות האסמבלי שבוצעו.

הקובץ leds.txt מכיל את סטטוס 32 הנורות. בכל מחזור שעון שאחת הנורות משתנה (נדלקת או נכבית), כותבים שורה עם שני מספרים ורווח ביניהם: המספר השמאלי הינו מחזור השעון, בדצימאלי, והמספר הימני מצב כל 32 הנורות ב- 8 ספרות הקסאדצימליות.

הקובץ monitor.txt מכיל את ערכי הפיסקלים שבמסך בסיום הריצה. כל שורה מכילה ערך פיסקל בודד (8 ביטים) בשתי ספרות הקסאדצימליות, כאשר סריקת המסך היא מלמעלה למטה, ומשמאל לימין. לדוגמא השורה הראשונה בקובץ מכילה את ערך הפיקסל בצד שמאל למעלה, והשורה האחרונה את הפיקסל בצד ימין למטה.

הקובץ monitor.txt הינו קובץ בינארי אשר מכיל את אותו דאטא כמו monitor.txt, וניתן אחרבץ שר מסד באמצעות התוכנה yuvplayer:

https://github.com/Tee0125/yuvplayer

size = 352x288 כאשר בפרמטרים בוחרים

הקובץ diskout.txt הינו קובץ פלט, באותו פורמט כמו memin.txt, שמכיל את תוכן הדיסק הקובץ הריצה.

האסמבלר

כדי שיהיה נוח לתכנת את המעבד וליצור את תמונות זיכרונות ההוראות והנתונים בקבצים imemin.txt ו- dmemin.txt, נכתוב בפרויקט גם את תוכנית האסמבלר. האסמבלר יכתב בשפת סי, ויתרגם את תוכנית האסמבלי שכתובה בטקסט בשפת אסמבלי, לשפת המכונה. ניתן להניח שקובץ הקלט תקין.

בדומה לסימולטור, האסמבלר הינו command line application עם שורת ההרצה הבאה:

asm.exe program.asm imem.txt dmem.txt

imem.txt , dmem.txt מכיל את תוכנית האסמבלי, וקבצי הפלט program.asm קובץ הקלט מכילים את תמונות זיכרונות ההוראות והנתונים. קבצי הפלט של האסמבלר משמשים אחייכ כקבצי הקלט של הסימולטור.

כל שורת קוד בקובץ האסמבלי מכילה את כל 5 הפרמטרים בקידוד ההוראה, כאשר הפרמטר הראשון הינו האופקוד, והפרמטרים מופרדים ע"י סימני פסיק. לאחר הפרמטר האחרון מותר להוסיף את הסימן # והערה מצד ימין, לדוגמא:

בכל הוראה, יש שלוש אפשרויות עבור שדה ה- immediate:

- ניתן לשים שם מספר דצימלי, חיובי או שלילי.
- . ניתן לשים מספר הקסאדצימלי שמתחיל ב- 0 ואז ספרות הקסאדצימליות.
- ניתן לשים שם סימבולי (שמתחיל באות). במקרה זה הכוונה ל- label, כאשר label מוגדר בקוד עייי אותו השם ותוספת נקודותיים.

: דוגמאות

L3:

jal \$imm, \$zero, \$zero, L4 # function call L4, save return addr in \$ra

halt \$zero, \$zero, \$zero, 0 # halt execution

L4:

beq \$ra, \$zero, \$zero, 0 # return from function in address in \$ra (reg1=0)

כדי לתמוך ב- labels האסמבלר מבצע שני מעברים על הקוד. במעבר הראשון זוכרים את הכתובות של כל ה- labels, ובמעבד השני בכל מקום שהיה שימוש ב- label בשדה ה- הכתובות של כל ה- label, מחליפים אותו בכתובת ה- label בפועל כפי שחושב במעבר הראשון. כמו כן שימו לב לשימוש ברגיסטר המיוחד \$imm בהוראות השונות. למשל הוראת ה- beq בדוגמא קופצת במידה ואפס שווה לאפס. תנאי זה מתקיים תמיד ולכן זו בעצם שיטה לממש

.unconditional jump

בנוסף להוראות הקוד, האסמבלר תומך בהוראה נוספת המאפשרת לקבוע תוכן של מילה 32 ישירות בתמונת זיכרון הנתונים.

.word address data

כאשר address הינו כתובת המילה ו- data תוכנה. כל אחד משני השדות יכול להיות בדצימלי, או משל: הקסאדצימלי בתוספת 0x. למשל:

.word 256 1 # set DMEM[256] = 1

.word 0x100 0x1234ABCD # DMEM[0x100] = DMEM[256] = 0x1234ABCD

הנחות נוספות

ניתן להניח את ההנחות הבאות:

- 1. ניתן להניח שאורך השורה המקסימאלי בקבצי הקלט הוא 500.
 - .2 ניתן להניח שאורך ה- label המקסימאלי הוא 50.
- 3. פורמט ה- label מתחיל באות, ואחייכ כל האותיות והמספרים מותרים.
- 4. צריך להתעלם מ- whitespaces כגון רווח או טאב. מותר שיהיו מספר רווחים או טאבים ועדיין הקלט נחשב תקין.
 - .upper case וגם lower case .5
 - 6. יש לעקוב אחרי שאלות, תשובות ועדכונים לפרויקט בפורום הקורס במודל.