Vahe Karamian
Computer Science 365
Computer Organization and Design
Project 3

Building An Assembler For JARC

This program should read in a JARC program, in text format, and translate each instruction into the equivalent binary instruction. Since JARC takes 16 bit instructions and the EPROMs output 8-bits, two EPROMs need to be programmed. The binary instructions are then saved in two binary files, one for the upper 8-bits and the other for the lower 8-bits. Each instruction has a similar format to that of the MIPS processor. Each possible input (RD, RS, RT, immediate, RAM address and ROM address) is multiplexed across 11 bits of the instruction. The attached sheet shows the 5-bit opcode for each instruction and the possible inputs. Because of the multiplexing of inputs, some bits are not used on certain instructions. These bits can be set to either one or zero. Even though these bits can be either one or zero, it is best to set them to one when using and EPROM. This helps prolong the life of the EPROM when it is erased. These examples help to better understand the binary translation.

XOR 0, 1, 3

This instruction takes the content of register 1 and 3 and exclusive ORs them together. The result is stored in register 0.

- Step 1: Find the opcode – XOR = 01100b
- Step 2: Determine what the inputs are – XOR takes RD< RS, and RT in that order. This means RD=0, RS=1, and RT=3.
- Step 3: Translate each input into the proper binary number. Note that different inputs have different lengths:
    - RD, RS and RT are each 2-bits
    - An immediate number is 4-bits
    - The ROM address and RAM address are each 7-bits

  RD=00b, RS = 01b, and RT = 11b for our example

- Step 4: Combine the known data in the proper format based on the attached sheet.
- Step 5: Divide the final instruction into two 8-bit numbers – 01100000b and 11111111b
- Step 6: Output the first number into one binary file and the second number into another binary file. Each subsequent instruction is added to the appropriate binary file. Click Here for Table.

Suppose the next instruction is BEQ 3, 2, 73

This means that if register 3 and register 2 are equal, the program will branch to line 73 of the program. We find that RS=11, RT=10b, and ROM Address = 1001001b and the opcode is 10011b. Note that when a ROM address is used the 7-bits are divided. The upper two bits are after the opcode and the last 5 bits are at the end. Click Here for Table.

		PROGRAM ROM 1										PROGRAM ROM 0						
							ROM							ROM ADDRESS				
												RAM ADDRESS						
												Immediate						
		Operation Code					RD		RS			RT						
Instruction		OP4	OP3	OP2	OP1	OP0	RD1	RD0	RS1	RS0		RT1	RT0	IM1	IM0	AD2	AD1	AD0
XOR	XOR RD, RS, RT	0	1	1	0	0	RD		RS			RT						
	XOR 0, 1, 3	0	1	1	0	0	0	0	0	1		1	1	x	x	x	x	x

		PROGRAM ROM 1										PROGRAM ROM 0						
							ROM							ROM ADDRESS				
												RAM ADDRESS						
												Immediate						
		Operation Code					RD		RS			RT						
Instruction		OP4	OP3	OP2	OP1	OP0	RD1	RD0	RS1	RS0		RT1	RT0	ROM4	ROM3	ROM2	ROM1	ROM0
BEQ	BEQ RD, RS, ROM	1	0	0	1	1	RD		RS			RT		ROM ADDRESS				
	BEQ 3, 2, 72	1	0	0	1	1	1	0	1	1		1	0	0	1	0	0	1

		PROGRAM ROM 1										**PROGRAM ROM 0**						
							ROM							**ROM ADDRESS**				
												RAM ADDRESS						
												Immediate						
		Operation Code					**RD**		**RS**			**RT**						
Instruction		**OP4**	**OP3**	**OP2**	**OP1**	**OP0**	**RD1**	**RD0**	**RS1**	**RS0**		**RT1**	**RT0**	**IM1**	**IM0**	**AD2**	**AD1**	**AD0**
ADD	ADD RD, RS, RT	0	0	0	0	0	RD		RS			RT						
		0	0	0	0	0												
ADDI	ADDI RD, RS, IMM	0	0	0	0	1	RD		RS			RT		IM1	IM0	AD2	AD1	AD0
		0	0	0	0	1												
ADDM	ADDM RD, RS, RAM	0	0	0	1	0	RD		RS			RT		IM1	IM0	AD2	AD1	AD0
		0	0	0	1	0												
SUB	SUB RD, RS, RT	0	0	0	1	1	RD		RS			RT		IM1	IM0	AD2	AD1	AD0
		0	0	0	1	1												
SUBI	SUBI RD, RS, IMM	0	0	1	0	0	RD		RS			RT		IM1	IM0	AD2	AD1	AD0
		0	0	1	0	0												
SUBM	SUBM RD, RS, RAM	0	0	1	0	1	RD		RS			RT		IM1	IM0	AD2	AD1	AD0
		0	0	1	0	1												
OR	OR RD, RS, RT	0	0	1	1	0	RD		RS			RT		IM1	IM0	AD2	AD1	AD0

		0	0	1	1	0										
ORI	ORI RD, RS, IMM	0	0	1	1	1	RD	RS	RT	IM1	IM0	AD2	AD1	AD0		
		0	0	1	1	1										
ORM	ORM RD, RS, RAM	0	1	0	0	0	RD	RS	RT	IM1	IM0	AD2	AD1	AD0		
		0	1	0	0	0										
AND	AND RD, RS, RT	0	1	0	0	1	RD	RS	RT	IM1	IM0	AD2	AD1	AD0		
		0	1	0	0	1										
ANDI	ANDI RD, RS, IMM	0	1	0	1	0	RD	RS	RT	IM1	IM0	AD2	AD1	AD0		
		0	1	0	1	0										
ANDM	ANDM RD, RS, RAM	0	1	0	1	1	RD	RS	RT	IM1	IM0	AD2	AD1	AD0		
		0	1	0	1	1										
XOR	XOR RD, RS, RT	0	1	1	0	0	RD	RS	RT	IM1	IM0	AD2	AD1	AD0		
		0	1	1	0	0										
XORI	XORI RD, RS, IMM	0	1	1	0	1	RD	RS	RT	IM1	IM0	AD2	AD1	AD0		
		0	1	1	0	1										
XORM	XORM RD, RS, RAM	0	1	1	1	0	RD	RS	RT	IM1	IM0	AD2	AD1	AD0		
		0	1	1	1	0										
NOT	NOT RD, RS, RT	0	1	1	1	1	RD	RS	RT	IM1	IM0	AD2	AD1	AD0		
		0	1	1	1	1										
SLT	SLT RD, RS, RT	1	0	0	0	0	RD	RS	RT	IM1	IM0	AD2	AD1	AD0		
		1	0	0	0	0										
SLTI	SLTI RD, RS, IMM	1	0	0	0	1	RD	RS	RT	IM1	IM0	AD2	AD1	AD0		
		1	0	0	0	1										
SLTM	SLTM RD, RS, RAM	1	0	0	1	0	RD	RS	RT	IM1	IM0	AD2	AD1	AD0		
		1	0	0	1	0										
BEQ	BEQ RD, RS, ROM	1	0	0	1	1	RD	RS	RT	IM1	IM0	AD2	AD1	AD0		
		1	0	0	1	1										
BNE	BNE RD, RS, ROM	1	0	1	0	0	RD	RS	RT	IM1	IM0	AD2	AD1	AD0		
		1	0	1	0	0										
J	J ROM	1	0	1	0	1	RD	RS	RT	IM1	IM0	AD2	AD1	AD0		
		1	0	1	0	1										
LWM	LWM RD, RAM	1	0	1	1	0	RD	RS	RT	IM1	IM0	AD2	AD1	AD0		
		1	0	1	1	0										
SWM	SWM RS, RAM	1	0	1	1	1	RD	RS	RT	IM1	IM0	AD2	AD1	AD0		
		1	0	1	1	1										
LWI	LWI RD, IMM	1	1	0	0	0	RD	RS	RT	IM1	IM0	AD2	AD1	AD0		
		1	1	0	0	0										

LWS	LWS RD,	1	1	0	0	1	RD	RS	RT	IM1	IM0	AD2	AD1	AD0
		1	1	0	0	1								
SHOW	SHOW RS,	1	1	0	1	0	RD	RS	RT	IM1	IM0	AD2	AD1	AD0
		1	1	0	1	0								

		PROGRAM ROM 1					PROGRAM ROM 0										
						ROM						ROM ADDRESS					
											RAM ADDRESS						
											Immediate						
		Operation Code					RD		RS		RT						
	Instruction	OP4	OP3	OP2	OP1	OP0	RD1	RD0	RS1	RS0	RT1	RT0	IM1	IM0	AD2	AD1	AD0
	lwi 0 5	1	1	0	0	0	0	0	1	1	0	1	0	1	1	1	1
	lwi 1 2	1	1	0	0	0	0	1	1	1	0	0	1	0	1	1	1
	swm 0 30	1	0	1	1	1	1	1	0	0	0	0	1	1	1	1	0
	subi 2 0 1	0	0	1	0	0	1	0	0	0	0	0	0	1	1	1	1
	beq 0 1 12	1	0	0	1	1	0	0	0	0	0	1	0	1	1	0	0
	j 6	1	0	1	0	1	0	0	1	1	1	1	0	0	1	1	0
	show 0	1	1	0	1	0	1	1	0	0	1	1	1	1	1	1	1
	andm 2 1 30	0	1	0	1	1	1	0	0	1	0	0	1	1	1	1	0
	show 2	1	1	0	1	0	1	1	1	0	1	1	1	1	1	1	1
	orm 2 1 30	0	1	0	0	0	1	0	0	1	0	0	1	1	1	1	0
	add 2 1 0	0	0	0	0	0	1	0	0	1	0	0	1	1	1	1	1
	xor 2 2 1	0	1	1	0	0	1	0	1	0	0	1	1	1	1	1	1
	not 3 2	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
	show 3	1	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1

Vahe Karamian
Computer Science 365
Project #3 - JARC Assembler

PROM1	PROM2
C1	AF
C3	97
BE	1E
24	0F
98	2C
A9	E6
D6	7F
5C	9E
D7	7F
44	9E
4	9F
65	3F
7F	7F
D7	FF