ECE 375
Computer Organization and Assembly Language Programming
Winter 2022
Assignment #4

The following questions are based on the enhanced AVR datapath (see Figures 8.24 and 8.26 in the text). The microoperation for the Fetch cycle is shown below.

Stage	Micro-operations
IF	IR ← M[PC], PC ← PC + 1, NPC ← PC + 1, RAR ← PC + 1

[25 pts]

1- Consider the implementation of the `MOVW Rd, Rr` (*Copy Register Word*) instruction on the enhanced AVR datapath.
   (a) List and explain the sequence of microoperations required to implement `MOVW Rd, Rr`.
   (b) List and explain the control signals and the Register Address Logic (RAL) output for the `MOVW Rd, Rr` instruction.

   Note that this instruction takes one execute cycle (EX). Control signals for the Fetch cycle are given below. Clearly explain your reasoning.

Control Signals	IF	MOVW EX
MJ	0	X
MK	0	X
ML	0	X
IR_en	1	X
PC_en	1	0
PCh_en	0	0
PCl_en	0	0
NPC_en	1	X
SP_en	0	X
DEMUX	x	X
MA	x	X
MB	x	X
ALU_f	xxxx	XXXX
MC	xx	01
RF_wA	0	1
RF_wB	0	1
MD	x	X
ME	x	X
DM_r	x	X
DM_w	0	X
MF	x	X
MG	x	1
Adder_f	xx	11
Inc_Dec	x	X
MH	x	X
MI	x	X

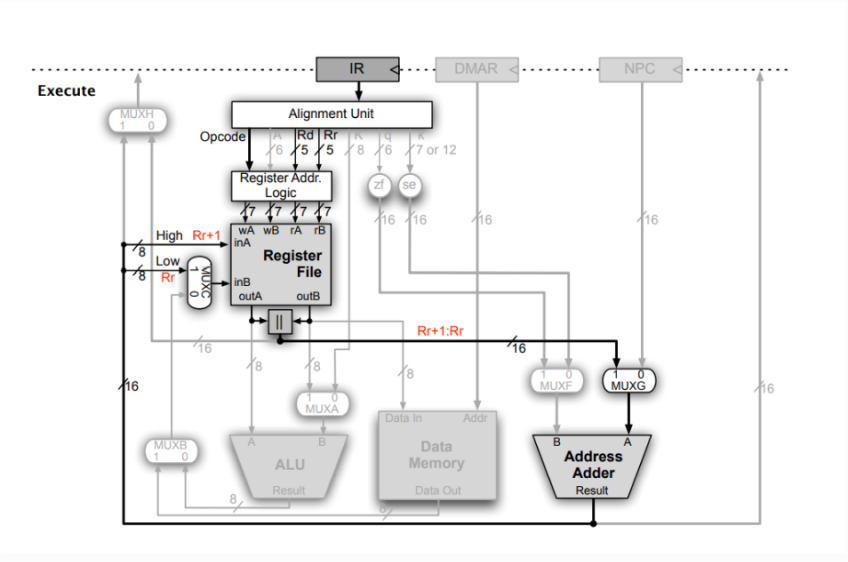
RAL Output	MOVW EX
wA	Rd+1
wB	Rd
rA	Rr+1
rB	Rr

(a) $Rd+1 : Rd \leftarrow Rr+1 : Rr$

Moves content in source register pairs to destination register pairs.

(a) Moves content in source register pairs to destination register pairs. It moves 16 bit content word from concatenated source register pairs to concatenated destination register pairs.

(b) Rr+1:Rr are outputed with in concatenated form Register File to Address adder, and then feedbacks to register file. To enable this processes, MuxG and Mux C should select 1, and address adder need to pass the address without in/decrement(Control signal: 11). Moreover, for writing content in destintion register pairs, RF_wA and RF_wB should be set.



[25 pts]

2- Consider the implementation of the `ST -X, Rr` (*Store Indirect and Pre-Decrement*) instruction on the enhanced AVR datapath.

(a) List and explain the sequence of microoperations required to implement `ST -X, Rr`.

(b) List and explain the control signals and the Register Address Logic (RAL) output for the `ST -X, Rr` instruction.

Note that this instruction takes two execute cycles (EX1 and EX2). Control signals for the Fetch cycle are given below. Clearly explain your reasoning.

Control Signals	IF	ST -X, Rr	
		EX1	EX2
MJ	0	X	X
MK	0	X	X
ML	0	X	X
IR_en	1	0	X
PC_en	1	0	0
PCh_en	0	0	0
PCl_en	0	0	0
NPC_en	1	X	X
SP_en	0	0	0
DEMUX	x	X	X
MA	x	X	X
MB	x	X	X
ALU_f	xxxx	XXXX	XXXX
MC	xx	XX	XX
RF_wA	0	0	0
RF_wB	0	0	0
MD	x	X	1
ME	x	X	1
DM_r	x	X	0
DM_w	0	0	1
MF	x	X	X
MG	x	1	X
Adder_f	xx	10	XX
Inc_Dec	x	X	X
MH	x	1	X
MI	x	X	X

RAL Output	ST -X, Rr	
	EX1	EX2
wA	X	X
wB	X	X
rA	XH	X
rB	XL	Rr

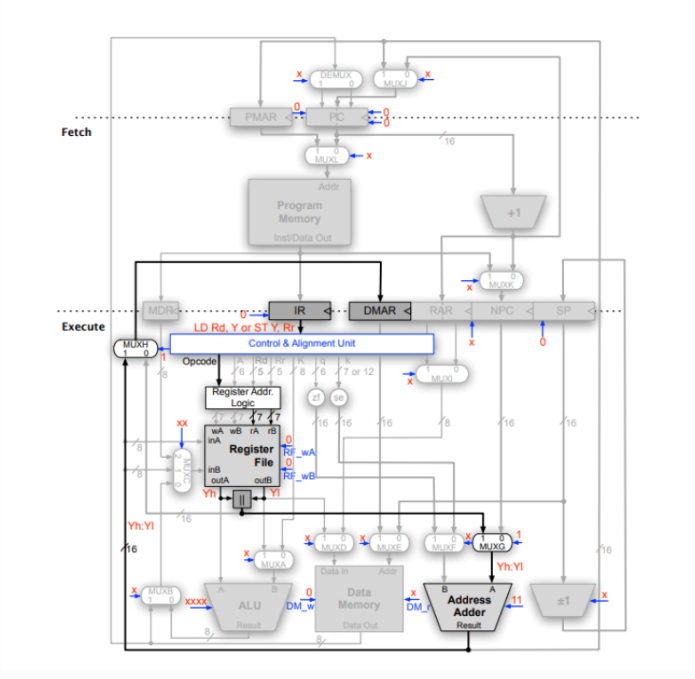
(a)

EX1) DMAR ← XH:XL − 1

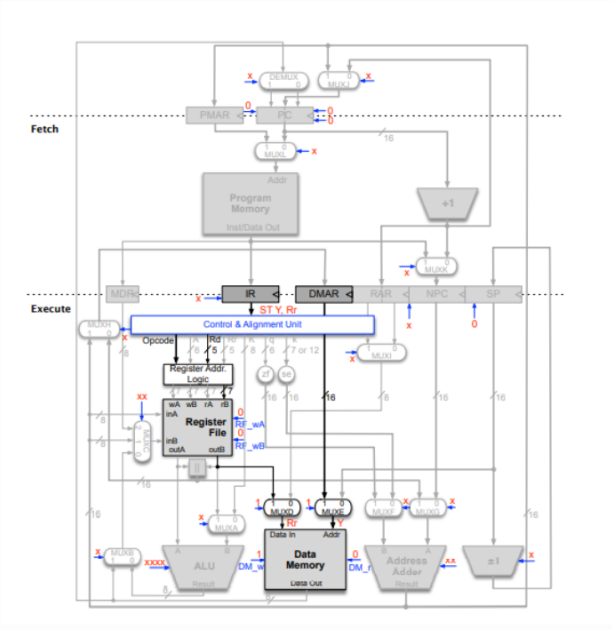
XH:XL ← XH:XL − 1

EX2) DM[MAR] ← Rr

(a)
First, X register is decremented and then copied to DMAR. Then, the content of register is stored in DM[DMAR]
(b)
In EX1, X register has to be decremented and then loaded to DMAR. For this reason, Mux H and Mux G have to select 1, and then Address adder has to decrement the X register (Control Signal: 10).
Other registers and Muxs have 0 or X values.



In EX2, the contnet of source register has to stored in data memory where X register is pointing.



For this reason, Mux D and Mux E have to select 1, and the source register has to be in rB in RA L output. Moreover, the Data memory has to write the content of source register so that DM_w is 1. The rest of the registers and multiplexers are x(don't care) or 0 values.

[25 pts]

3- Consider the implementation of the `ICALL` (Indirect Call to Subroutine) instruction on the enhanced AVR datapath. `ICALL` is similar to the `RCALL` (Relative Call to Subroutine) instruction, except that the Z register points to the target address.

(a) List and explain the sequence of microoperations required to implement `ICALL`.

(b) List and explain the control signals and the Register Address Logic (RAL) output for the `ICALL` instruction Note that this instruction takes two execute cycles (EX1 and EX2). Control signals for the Fetch cycle are given below. Clearly explain your reasoning

Control Signals	IF	ICALL	
		EX1	EX2
MJ	0	X	1
MK	0	X	X
ML	0	X	X
IR_en	1	0	X
PC_en	1	X	1
PCh_en	0	0	0
PCl_en	0	0	0
NPC_en	1	0	X
SP_en	0	1	1
DEMUX	x	X	X
MA	x	X	X
MB	x	X	X
ALU_f	xxxx	XXXX	XXXX
MC	xx	X X	X X
RF_wA	0	0	0
RF_wB	0	0	0
MD	x	0	0
ME	x	0	0
DM_r	x	0	0
DM_w	0	1	1
MF	x	X	X
MG	x	X	1
Adder_f	xx	X X	11
Inc_Dec	x	1	1
MH	x	X	X
MI	x	0	1

RAL Output	ICALL	
	EX1	EX2
wA	X	X
wB	X	X
rA	X	ZH
rB	X	ZL

a)

EX1) DM[SP] ← RAR(L), SP ← SP-1

EX2) DM[SP] ← RAR(H), SP ← SP-1, PC ← Z

(a)
Assumption: Z register is already pointing target address.
The return address is PC+1, and this is stored in RAR.
In EX1, the lower byte of RAR is pushed to stack.
In EX2, the higher byte of RAR is pushed to stack, and then PC points the target address stored in Z register.

(b)
In Ex1, the lower byte of RAR has to be written in Data memory with Stack pointer. For this reason, Mux I, Mux D, and Mux E have to select 0, and Inc/Dec has to be 1 in order to decrement stack pointer. Moreover, DM_w has to be set. Other registers or Muxs have 0 or X values.

In Ex2, the higher byte of RAR has to be written in Data memory with Stack pointer. For this reason, Mux I, Mux D, and Mux E have to select 0, and Inc/Dec has to be 1 in order to decrement stack pointer. Moreover, DM_w has to be set. The content of Z-register has to be loaded to PC. ZH:ZL has to loaded to address adder and then sent to PC. Thus, Mux G and Mux J have to slect 1.

Other registers or Muxs have 0 or X values.

[25 pts]

4- Consider the implementation of the `LPM` (*Load Program Memory*) instruction on the enhanced AVR datapath.
(a) List and explain the sequence of microoperations required to implement `LPM`.
(b) List and explain the control signals and the Register Address Logic (RAL) output for the `LPM` instruction. Note that this instruction takes three execute cycles (EX1, EX2, and EX3). Control signals for the Fetch cycle are given below. Clearly explain your reasoning.

Control Signals	IF	LPM		
		EX1	EX2	EX3
MJ	0	x	x	x
MK	0	x	x	x
ML	0	x	1	x
IR_en	1	x	x	x
PC_en	1	0	0	0
PCh_en	0	0	0	0
PCl_en	0	0	0	0
NPC_en	1	x	x	x
SP_en	0	0	0	0
DEMUX	x	x	x	x
MA	x	x	x	x
MB	x	x	x	x
ALU_f	xxxx	xxxx	xxxx	xxxx
MC	xx	xx	xx	10
RF_wA	0	0	x	0
RF_wB	0	0	x	1
MD	x	x	x	x
ME	x	x	x	x
DM_r	x	x	x	x
DM_w	0	x	x	x
MF	x	x	x	x
MG	x	1	x	x
Adder_f	xx	11	xx	xx
Inc_Dec	x	x	x	x
MH	x	x	x	x
MI	x	x	x	x

RAL Output	LPM		
	EX1	EX2	EX3
wA	x	x	x
wB	x	x	R0
rA	ZH	x	x
rB	ZL	x	x

a)

EX1) PMAR ← Z

EX2) MDR ← PM[PMAR]

EX3) R0 ← MDR

(a)
Instruction LPM implies loading one byte data; R0 <- (Z)
Assumption: Z - register is already pointing program memory address.

In Ex1, similar to LD instruction, PMAR has to be loaded from address register, Z-register.
In Ex2, PM[PMAR] has to be in the process of being input of register file. Thus, the content of PM[PMAR] is temporarily in MDR.
In EX3, the content of MDR has to be written in R0 in register file.

(b)
In Ex1, the content of Z-register has to feed in PMAR. Thus, MUX G has to select 1, and Address adder pass the address without in/decrement and send it to PMAR. Except these two, others are X or 0 values.
In Ex2, the PM[PMAR] has to be a output of Program memory and loaded to MDR. For these reasons, MUX L has to choose 1 to put PMAR into Program memory. Except Mux L, others have X or 0 values.
In Ex3, the content of MDR has to written in R0. Thus, Mux C slects 10(2), and RF_wB is enabled because R0 is loaded wB. Other registers or Muxs have 0 or X values.