ECE 375 Homework 4

Computer Organization and Assembly Language Programming

Winter Term 2018

Jeremy Fischer

1 Homework Questions

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 $\leftarrow$ M[PC], PC $\leftarrow$ PC + 1, NPC $\leftarrow$ PC + 1, RAR $\leftarrow$ PC + 1

1. Consider the implementation of the *CLR Rd* (Clear Register) instruction on the enhanced AVR datapath.

   a List and explain the sequence of microoperations required to implement *CLR Rd*. Note that this instruction takes a single execute cycle (EX).

   b List and explain the control signals and the Register Address Logic (RAL) output for the *CLR Rd* instruction. Control signals for the Fetch cycle are given below. Clearly explain your reasoning.

   Answer:

   Rd $\leftarrow$ Rd $\oplus$ Rd
   Simply EOR the source register, Rd, with itself.

   - MA = 1 due to Rd duplicate coming from the || operator.
   - ALU_f = 1010 for EOR
   - RF_wB = 1 due to writing to wB (Rd)
   - DM_w = 0 to disallow writing to Data Memory

CLR		
Control Signals	IF	EX
MJ	0	x
MK	0	x
ML	0	x
IR_en	1	x
PC_en	1	0
PCh_en	0	0
PCI_en	0	0
NPC_en	1	x
SP_en	0	0
DEMUX	x	x
MA	x	1
MB	x	0
ALU_f	xxxx	1010
MC	xx	00
RF_wA	0	0
RF_wB	0	1
MD	x	x
ME	x	x
DM_r	x	x
DM_w	0	0
MF	x	x
MG	x	x
Adder_f	xx	xx
Inc_Dec	x	x
MH	x	x
MI	x	x

CLR	
RAL output	EX
wA	x
wB	Rd
rA	x
rB	Rd

2. Consider the implementation of the *STD Y+q, Rr* (Store Indirect with Displacement) instruction on the enhanced AVR datapath.

   a List and explain the sequence of microoperations required to implement *STD Y+q, Rr*. Note that this instruction takes two execute cycles (EX1 and EX2).

   b List and explain the control signals and the Register Address Logic (RAL) output for the *STD Y+q, Rr* instruction.

   Control signals for the Fetch cycle are given below. Clearly explain your reasoning.

   Answer:

   First cycle: Get the address to store Rr, (Y+q)
   Second cycle: then store Rr there.

   DMAR $\leftarrow$ AR + q
   M[DMAR] $\leftarrow$ Rr

   - In the first execution cycle we only need to get YH:YL and then add it to q inside of the Address Adder and output the result to DMAR.
   - In the second cycle we only need to read Rr, and store it in M[DMAR].
   - You can see the above take place in the RAL table and EX1/EX2 columns in the control signals table.

STD Y+q, Rr			
Control Signals	IF	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
PCI_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	1	x
MG	x	1	x
Adder_f	xx	00	xx
Inc_Dec	x	x	x
MH	x	1	x
MI	x	x	x

STD Y+q, Rr		
RAL output	EX1	EX2
wA	x	x
wB	x	x
rA	YH	Rr
rB	YL	x

3. Consider the implementation of the *RCALL k* (*Relative Call to Subroutine*) instruction on the enhanced AVR datapath.

   a List and explain the sequence of microoperations required to implement *RCALL k*. Note that this instruction takes two execute cycles (EX1 and EX2).

   b List and explain the control signals and the Register Address Logic (RAL) output for the *RCALL k* instruction

   Control signals for the Fetch cycle are given below. Clearly explain your reasoning.

   Answer:

   First cycle: RAR gets PC + 1, PC points to (PC + k)
   Second cycle: Jump to (PC + k)

   NPC ← M[PC], RAR ← PC + 1, M[SP] ← RARL, SP ← SP - 1
   PC ← NPC, M[SP] ← RARL, SP ← SP - 1

RCALL			
Control Signals	IF	EX1	EX2
MJ	0	0	1
MK	0	0	x
ML	0	0	0
IR_en	1	1	x
PC_en	1	1	x
PCh_en	0	0	0
PCl_en	0	0	0
NPC_en	1	1	0
SP_en	0	1	1
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	0	0
ME	x	0	0
DM_r	x	x	x
DM_w	0	1	1
MF	x	0	x
MG	x	0	x
Adder_f	xx	00	xx
Inc_Dec	x	1	1
MH	x	x	x
MI	x	0	1

RCALL		
RAL output	EX1	EX2
wA	x	x
wB	x	x
rA	x	x
rB	x	x

4. Consider the implementation of the *LPM R16, Z+* (Load Program Memory) instruction on the enhanced AVR datapath.

   a List and explain the sequence of microoperations required to implement *LPM R16, Z+*. Note that this instruction takes three execute cycles (EX1, EX2, and EX3).

   b List and explain the control signals and the Register Address Logic (RAL) output for the LPM instruction. Control signals for the Fetch cycle are given below. Clearly explain your reasoning.

   Control signals for the Fetch cycle are given below. Clearly explain your reasoning.

   Answer:

   PMAR ← Z, PC ← PC + 1
   MDR ← PM[PMAR], ZH:ZL ← (ZH:ZL) + 1
   R16 ← MDR

   First cycle: Read the address in ZH:ZL., PC = PC + 1
   After the first cycle PMAR will have ZH:ZL

   Second cycle: MDR ← PM[ZH:ZL], Z ← Z + 1
   After the second cycle MDR will have PM[Z], and Z will have been incremented and stored.

   Third cycle: R16 ← MDR
   After the third cycle R16 will have the value of MDR

   - First cycle MUXL must be done to read ZH:ZL from PMAR
   - Second cycle MUXC must be 01 to inB will be reading ZL from the address adder
   - Third cycle MUXC must be 10 to so inB will be MDR

LPM				
Control Signals	IF	EX1	EX2	EX3
MJ	0	x	x	x
MK	0	x	x	x
ML	0	x	1	x
IR_en	1	0	0	x
PC_en	1	0	0	0
PCh_en	0	0	0	0
PCI_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	01	10
RF_wA	0	0	1	0
RF_wB	0	0	1	1
MD	x	x	x	x
ME	x	x	x	x
DM_r	x	x	x	x
DM_w	0	0	0	0
MF	x	x	x	x
MG	x	1	1	x
Adder_f	xx	11	01	xx
Inc_Dec	x	x	x	x
MH	x	1	x	x
MI	x	x	x	x

LPM			
RAL output	EX1	EX2	EX3
wA	x	ZH	x
wB	x	ZL	R16
rA	ZH	ZH	x
rB	ZL	ZL	x