ECE 375
Computer Organization and Assembly Language Programming
Winter 2015
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 | |
| MK | 0 | |
| ML | 0 | |
| IR_en | 1 | |
| PC_en | 1 | |
| PCh_en | 0 | |
| PCl_en | 0 | |
| NPC_en | 1 | |
| SP_en | 0 | |
| DEMUX | x | |
| MA | x | |
| MB | x | |
| ALU_f | xxxx | |
| MC | xx | |
| RF_wA | 0 | |
| RF_wB | 0 | |
| MD | x | |
| ME | x | |
| DM_r | x | |
| DM_w | 0 | |
| MF | x | |
| MG | x | |
| Adder_f | xx | |
| Inc_Dec | x | |
| MH | x | |
| MI | x | |

| RAL Output | `MOVW` EX |
|---|---|
| wA | |
| wB | |
| rA | |
| rB | |

[25 pts]

2- Consider the implementation of the `LD Rd, -X` (*Load Indirect and Pre-decrement*) instruction on the enhanced AVR datapath.

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

(b) List and explain the control signals and the Register Address Logic (RAL) output for the `LD Rd, -X` instruction. Some of the control signals are given below.

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 | `LD Rd, -X` | |
|---|---|---|---|
| | | EX1 | EX2 |
| MJ | 0 | | |
| MK | 0 | | |
| ML | 0 | | |
| IR_en | 1 | | |
| PC_en | 1 | | |
| PCh_en | 0 | | |
| PCl_en | 0 | | |
| NPC_en | 1 | | |
| SP_en | 0 | | |
| DEMUX | x | | |
| MA | x | | |
| MB | x | | |
| ALU_f | xxxx | | |
| MC | xx | | |
| RF_wA | 0 | | |
| RF_wB | 0 | | |
| MD | x | | |
| ME | x | | |
| DM_r | x | | |
| DM_w | 0 | | |
| MF | x | | |
| MG | xx | | |
| Adder_f | xx | | |
| Inc_Dec | x | | |
| MH | x | | |
| MI | x | | |

| RAL Output | `LD Rd, -X` | |
|---|---|---|
| | EX1 | EX2 |
| wA | | |
| wB | | |
| rA | | |
| rB | | |

[25 pts]

3- Consider the implementation of the `ICALL` (Indirect Call to Subroutine) instruction on the enhanced AVR datapath shown below. `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 | | |
| MK | 0 | | |
| ML | 0 | | |
| IR_en | 1 | | |
| PC_en | 1 | | |
| PCh_en | 0 | | |
| PCl_en | 0 | | |
| NPC_en | 1 | | |
| SP_en | 0 | | |
| DEMUX | x | | |
| MA | x | | |
| MB | x | | |
| ALU_f | xxxx | | |
| MC | xx | | |
| RF_wA | 0 | | |
| RF_wB | 0 | | |
| MD | x | | |
| ME | x | | |
| DM_r | x | | |
| DM_w | 0 | | |
| MF | x | | |
| MG | xx | | |
| Adder_f | xx | | |
| Inc_Dec | x | | |
| MH | x | | |
| MI | x | | |

| RAL Output | ICALL | |
|---|---|---|
| | EX1 | EX2 |
| wA | | |
| wB | | |
| rA | | |
| rB | | |

[25 pts]

4- Consider the implementation of the `RET` (Return from subroutine) instruction on the enhanced AVR datapath shown on the following page.

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

(b) List and explain the control signals and the Register Address Logic (RAL) output for the `RET` instruction.

Note that this instruction takes three execute cycles (EX1, EX2, and EX3). The Fetch cycle is shown below. Clearly explain your reasoning.

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

| RAL Output | RET | | |
|---|---|---|---|
| | EX1 | EX2 | EX3 |
| wA | | | |
| wB | | | |
| rA | | | |
| rB | | | |