Computer Architecture Project - Phase 1

Team 1

Ahmad Khaled	S01 BN 03
Zeinab Rabie	S01 BN 26
Sara Maher	S01 BN 28
Abdelrahman Mohamed	S02 BN 03

Computer Engineering Department - Cairo University

1 Instruction Set Architecture

The structure for the IR for two-operand instructions is given in Figure 1. The structure for one-operand instructions is given in Figure 2. The structure for branching instructions is given in Figure 3. The structure for the rest of the instructions is given in Figure 4.

Direct	Register	Auto-increment	Auto-decrement	Indexed
Code	000	001	010	011
Indirect	Register	Auto-increment	Auto-decrement	Indexed
Code	100	101	110	111

Table 1: Addressing Mode Codes

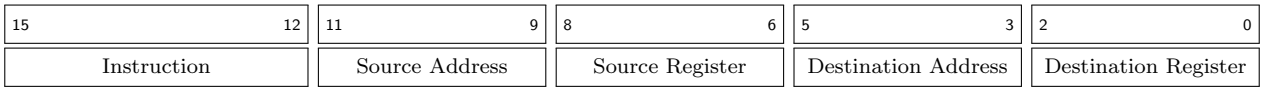
	R_0	R_1	R_2	R_3
Code	000	001	010	011
	R_4	R_5	R_6?	R_7
Code	100	101	110	111

Table 2: Register Addressing Codes

Instruction	MOV	ADD	ADC	SUB	SBC	AND	OR	XNOR	CMP
OP Code	1111	1110	1101	1100	1011	1010	1001	1000	0111

Table 3: Two-Operand Instruction Codes

Figure 1: IR Structure For Two-Operand Instructions



Instruction specifies the instruction to execute. Possible codes given in Table 3.

Source Address specifies the addressing mode to choose for the source. Possible codes given in Table 1.

Source Register specifies the source register. Possible codes given in Table 2.

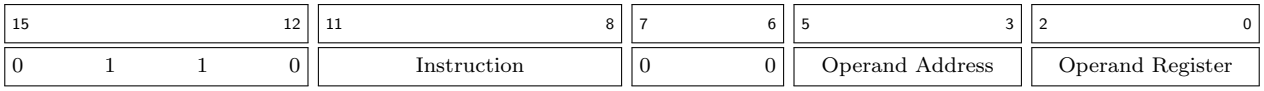
Destination Address specifies the addressing mode to choose for the destination. Possible codes given in Table 1.

Destination Register specifies the destination register. Possible codes given in Table 2.

Instruction	INC	DEC	CLR	INV	LSR	ROR	RRC	ASR	LSL	ROL	RLC
OP Code	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010

Table 4: Single Operand Instruction Codes

Figure 2: IR Structure For Single-Operand Instructions



Instruction specifies the instruction to execute. Possible codes given in Table 4.

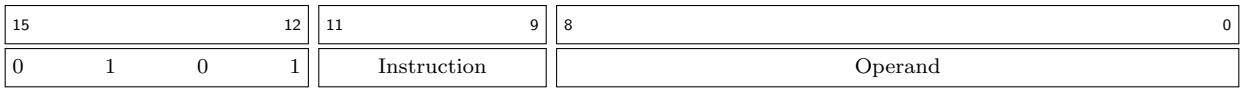
Operand Address specifies the addressing mode to choose for the operand. Possible codes given in Table 1.

Operand Register specifies the operand register. Possible codes given in Table 2.

Instruction	BR	BEQ	BNE	BLO	BLS	BHI	BHS
OP Code	000	001	010	011	100	101	110

Table 5: Branching Instruction Codes

Figure 3: IR Structure For Branching Instructions



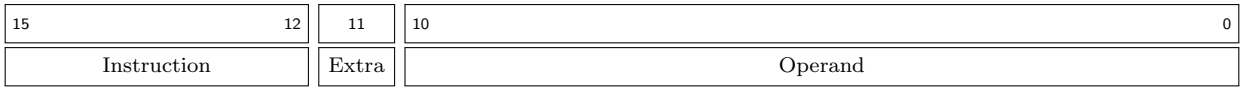
Instruction specifies the instruction to execute. Possible codes given in Table 5.

Operand specifies the branching offset.

Instruction	JSR	RTS	HITR	IRET	HLT	NOP
OP Code	0100	0011	0011	0010	0001	0000

Table 6: Miscellaneous Instruction Codes

Figure 4: IR Structure For Miscellaneous Instructions



Instruction specifies the instruction to execute (except for differentiating RTS and HITR, done using Extra bit). Possible codes given in Table 6.

Extra useful for JSR and HITR only. specifies whether the Operand is given directly or using a register/addressing mode for JSR. Indicates HITR when instruction is 0011 and Extra= 1, indicates RTS if instruction is 0011 and Extra= 0.

Operand useful for JSR only. specifies the memory location to jump to.

2 Bus System Schematic

Figure 5 includes the schematic for the microprocessor design. The actual register file contains all the registers, but the special registers (temporaries - MDR - MAR) are shown outside here for clarity.

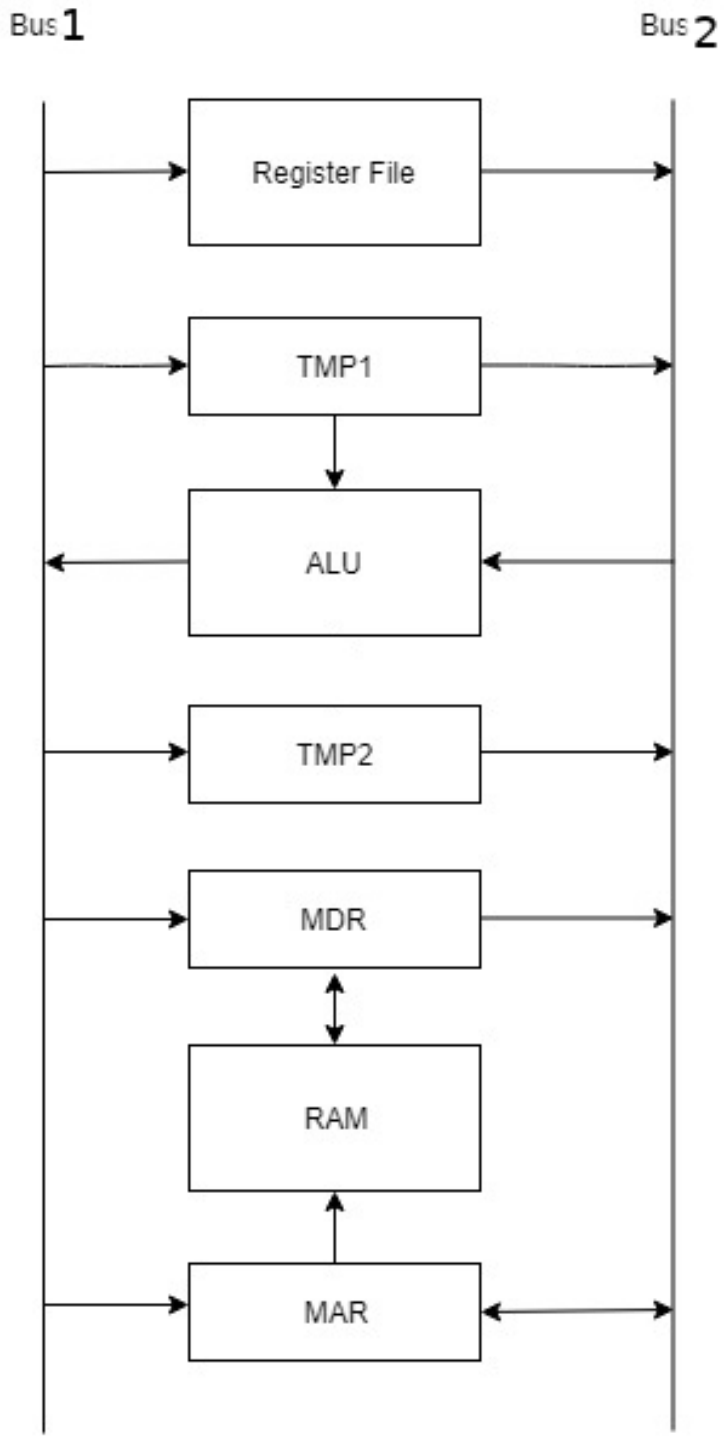


Figure 5: Bus System Schematic

3 Micro-Instructions

3.1 Control Signals

We analyze the Fetch / Execution phase each on its own and mention the MA / Clock Cycles that are associated only with this phase. See the next subsection for the total analysis. The $F = X$ instructions control the ALU's operations. We use the same convention as the textbook (Hamacher, Vranseic, Zaaky - Computer Organization).

1. **Fetch and Decode**. MA = 1, Clock Cycles = 2.
   (a) PC_{out}, MAR^1_{in}, $F = A + 1$, PC_{in}, Rd
   (b) MDR_{out}, $F = A$, IR_{in}
2. **Fetch Op**. Please note that if two operands are desired, the result of the source operand (outed in the last line) is output to TMP2 register instead of TMP1 at the end, then the destination is fetched and placed in TMP1.
   (a) Op = R_i. MA = 0, Clock Cycles = 1.
      i. $R_{i\text{out}}$, $F = A$, TMP1_{in}
   (b) Op = $(R_i)+$. MA = 1, Clock Cycles = 2.
      i. $R_{i\text{out}}$, MAR^1_{in}, $F = A + 1$, $R_{i\text{in}}$, Rd, WMFC
      ii. MDR_{out}, $F = A$, TMP1_{in}
   (c) Op = $-(R_i)$. MA = 1, Clock Cycles = 2.
      i. $R_{i\text{out}}$, $F = A - 1$, MAR_{in}, $R_{i\text{in}}$, Rd, WMFC
      ii. MDR_{out}, $F = A$, TMP1_{in}
   (d) Op = $X(R_i)$. MA = 2, Clock Cycles = 4.
      i. PC_{out}, MAR^1_{in}, Rd, $F = A + 1$, TMP1_{in}, PC_{in}, WMFC
      ii. MDR_{out}, $F = A + B$, MAR_{in}, Rd, WMFC
      iii. MDR_{out}, $F = A$, TMP1_{in}
   (e) Op = $@R_i$. MA = 1, Clock Cycles = 2.
      i. $R_{i\text{out}}$, $F = A$, MAR_{in}, Rd, WMFC
      ii. MDR_{out}, $F = A$, TMP1_{in}
   (f) Op = $@(R_i)+$. MA = 2, Clock Cycles = 3.
      i. $R_{i\text{out}}$, MAR^1_{in}, Rd, $F = B + 1$, $R_{i\text{in}}$, WMFC
      ii. MDR_{out}, $F = A$, MAR_{in}, Rd, WMFC
      iii. MDR_{out}, $F = A$, TMP1_{in}
   (g) Op = $@-(R_i)$. MA = 2, Clock Cycles = 3.
      i. $R_{i\text{out}}$, $F = A - 1$, MAR_{in}, Rd, $R_{i\text{in}}$, WMFC
      ii. MDR_{out}, $F = A$, MAR_{in}, Rd, WMFC
      iii. MDR_{out}, $F = A$, TMP1_{in}
   (h) Op = $@X(R_i)$. MA = 3, Clock Cycles = 4.

i. PC_{out}, MAR^1_{in}, Rd, $F = A + 1$, PC_{in}, TMP1_{out}, WMFC
ii. MDR_{out}, $F = A + B$, MAR_{in}, Rd, WMFC
iii. MDR_{out}, $F = A$, MAR_{in}, Rd, WMFC
iv. MDR_{out}, $F = A$, TMP1_{in}

3. **Two-Operand Execution Instructions**. MA = 0 if Dst is a register, 1 otherwise. Clock Cycles = 1.
We assume that the Source operand is in TMP2 (connected to the ALU as B) and the Destination operand is in TMP1. Dst_{in} is replaced by "MDR_{in}, WT" if Dst is a memory location and is replaced by $R_{i\text{in}}$ if Dst is a register. The flags register is also set appropriately during this phase.

(a) MOV Src, Dst
i. TMP2_{out}, $F = A$, Dst_{in}
(b) ADD Src, Dst
i. TMP2_{out}, $F = A + B$, Dst_{in}
(c) ADC Src, Dst
i. TMP2_{out}, $F = A + B + C$, Dst_{in}
(d) SUB Src, Dst
i. TMP2_{out}, $F = A - B$, Dst_{in}
(e) SBC Src, Dst
i. TMP2_{out}, $F = A - B - C$, Dst_{in}
(f) AND Src, Dst
i. TMP2_{out}, $F = A$ AND B, Dst_{in}
(g) OR Src, Dst
i. TMP2_{out}, $F = A$ OR B, Dst_{in}
(h) XNOR Src, Dst
i. TMP2_{out}, $F = A$ XNOR B, Dst_{in}
(i) CMP Src, Dst
i. TMP2_{out}, $F = A - B$

4. **Single-Operand Instructions**. MA = 0 if Dst is a register, 1 otherwise. Clock Cycles = 1.
We assume that the destination operand is in TMP1. Dst_{in} is replaced by MDR_{in}, Write if Dst is a memory location and is replaced by $R_{i\text{in}}$ if Dst is a register. The flags register is also set appropriately during this phase.

(a) INC Dst
i. TMP1_{out}, $F = A + 1$, Dst_{in}
(b) DEC Dst
i. TMP1_{out}, $F = A - 1$, Dst_{in}
(c) CLR Dst

i. $F = 0$, Dst_{in}

(d) INV Dst

i. TMP1_{out}, $F = \text{INV}(A)$, Dst_{in}

(e) LSR Dst

i. TMP1_{out}, $F = \text{LSR}(A)$, Dst_{in}

(f) ROR Dst

i. TMP1_{out}, $F = \text{ROR}(A)$, Dst_{in}

(g) RRC Dst

i. TMP1_{out}, $F = \text{RRC}(A)$, Dst_{in}

(h) ASR Dst

i. TMP1_{out}, $F = \text{ASR}(A)$, Dst_{in}

(i) LSL Dst

i. TMP2_{out}, $F = \text{LSL}(A)$, Dst_{in}

(j) ROL Dst

i. TMP2_{out}, $F = \text{ROL}(A)$, Dst_{in}

(k) RLC Dst

i. TMP2_{out}, $F = \text{RLC}(A)$, Dst_{in}

5. **No Operand Instructions**. MA = 0, Clock Cycles = 1. Assume the existence of HLT/NOP control signals.

(a) HLT.

i. HLT

(b) NOP.

i. NOP

6. **Branching Instructions**. MA = 0, Clock Cycles = 2.

(a) BR

i. PC_{out}, $F = A$, TMP1_{in}

ii. Offset Part of IR_{out}, $F = A + B$, PC_{in}

(b) BEQ

i. PC_{out}, $F = A$, TMP1_{in}, If $Z = 1$ then End

ii. Offset Part of IR_{out}, $F = A + B$, PC_{in}

(c) BNEQ

i. PC_{out}, $F = A$, TMP1_{in}, If $Z = 0$ then End

ii. Offset Part of IR_{out}, $F = A + B$, PC_{in}

(d) BLO

i. PC_{out}, $F = A$, TMP1_{in}, If $C = 0$ then End

ii. Offset Part of IR_{out}, $F = A + B$, PC_{in}

(e) BLS

i. PC_{out}, $F = A$, TMP1_{in}, If $C = 0$ or $Z = 1$ then End

ii. Offset Part of IR_{out}, $F = A + B$, PC_{in}

(f) BHI

i. PC_{out}, $F = A$, TMP1_{in}, If $C = 1$ then End

ii. Offset Part of IR_{out}, $F = A + B$, PC_{in}

(g) BHS

i. PC_{out}, $F = A$, TMP1_{in}, If $C = 1$ or $Z = 1$ then End

ii. Offset Part of IR_{out}, $F = A + B$, PC_{in}

7. **Jump Instructions**.

(a) JSR. MA = 3, Clock Cycles = 7.

i. PC_{out}, $F = A$, MAR_{in}, TMP1_{in}, Rd

ii. $F = B + 1$, PC_{in}, WMFC

iii. MDR_{out}, $F = A$, TMP2_{in}

iv. SP_{out}, $F = A - 1$, TMP1_{in}

v. $F = B$, SP_{in}, MAR_{in}

vi. PC_{out}, $F = A$, MDR_{in}, Write, WMFC

vii. TMP2_{out}, $F = A$, PC_{in}

(b) RTS. MA = 2, Clock Cycles = 3.

i. SP_{out}, $F = A$, MAR_{in}, TMP2_{in}, Rd

ii. TMP2_{out}, $F = A + 1$, SP_{in}, WMFC

iii. MDR_{out}, $F = A$, PC_{in}

(c) HITR. MA = 3, Clock Cycles = 6.
Assuming the address to go to is stored in the IR.

i. SP_{out}, $F = A - 1$, TMP1_{in}

ii. TMP1_{in}, $F = A$, MAR_{in}, SP_{in}

iii. FLAG_{out}, $F = A$, MDR_{in}, Write, WMFC

iv. $F = B - 1$, MAR_{in}, SP_{in}

v. PC_{out}, $F = A$, MDR_{in}, Write, WMFC

vi. Address Part of IR_{out}, $F = A$, PC_{in}

(d) IRET. MA = 3, Clock Cycles = 6.

i. SP_{out}, $F = A$, MAR_{in}, TMP2_{in}, Read

ii. TMP2_{out}, $F = A + 1$, SP_{in}, TMP1_{in}, WMFC

iii. MDR_{out}, $F = A$, PC_{in}

iv. TMP1_{out}, MAR_{in}, Read

v. $F = B + 1$, SP_{in}, WMFC

vi. MDR_{out}, $F = A$, FLAG_{in}

3.2 Memory and Cycle Analysis

The average for all instructions is given in Table 7. The MA / Clock Cycles in this subsection include the cycles and accesses required for fetching and decoding as well as fetching operands when necessary. Note: This analysis is **OUTDATED**, the current average number of clock cycles is a LOWER (by about 1 cycle for one- and two-operand instructions).

Instruction	Two-Op (9)	Single-Op (11)	Zero-Op (2)	Branching (7)	Average (Excl Jumps)
Average MA	4.875	3.375	1	1	**3.1034**
Average CC	9.5	6.25	3	4	**6.4913**

Table 7: Average MA / Clock Cycles for design. The number between braces in the instruction name is the count of instructions belonging to that category.

The detailed analyses for two-operand instructions according to source / destination addressing modes are given in Table 8 for MA and in Table 9 for Clock Cycles. The detailed analysis for single-operand instructions according to operand address is given in Table 10. The Jumps are given in Table 11.

Src Dst	R_i	$-(R_i)$ / $(R_i)+$ / $@R_i$	$X(R_i)$ / $@(R_i)+$ / $@-(R_i)$	$@X(R_i)$
R_j	1	2	3	4
$-(R_j)$ $(R_j)+$ $@R_j$	3	4	5	6
$X(R_j)$ $@(R_j)+$ $@-(R_j)$	4	5	6	7
$@X(R_j)$	5	6	7	8

Table 8: Memory-Access Two-Operand Instruction Analysis.

Src Dst	R_i	$@R_i$	$-(R_i)$ / $(R_i)+$	$X(R_i)$ / $@(R_i)+$ / $@-(R_i)$	$@X(R_i)$
R_j	5	6	7	8	9
$@R_i$	8	7	8	9	10
$-(R_j)$ $(R_j)+$	7	8	9	10	11
$X(R_j)$ $@(R_j)+$ $@-(R_j)$	8	9	10	11	12
$@X(R_j)$	9	10	11	12	13

Table 9: Clock Cycles Two-Operand Instruction Analysis.

Op	R_i	$@R_i$	$-(R_i)$ / $(R_i)+$	$X(R_i)$ / $@(R_i)+$ / $@-(R_i)$	$@X(R_i)$
MA	1	3	3	4	5
Clock Cycles	4	5	6	7	8

Table 10: MA and Clock Cycle Instruction Analysis for Single Operand Instructions

Instruction	JSR	RTS	HITR	IRET	Average
MA	4	3	3	4	3.5
Clock Cycles	9	5	5	8	6.75

Table 11: MA and Clock Cycle Instruction Analysis for Jump Instructions