Computer Architecture Project

Phase One

Team #10

Name	Section	BN
Khaled Galal	1	17
Muhamed Ayman	2	11
Yousif Gamal	2	36
Youssef Mohamed	2	37

OP Codes

2-Operands Operations

$IR_{15\ ->12}$

0 0 1 0 AND

0 0 1 1 OR

1 0 1 0 Xnor

1 0 1 1 CMP

0 1 0 0 ADD

0 1 0 1 ADC

1 1 0 0 SUB

1 1 0 1 SBC

0 1 1 0 MOV

1-Operand Operations

$IR_{15\ ->6}$

0 0 0 0 0 0 0 0 0 1 INC

0 0 0 0 0 0 0 0 1 0 DEC

0 0 0 0 0 0 0 0 1 1 CLR

0 0 0 0 0 0 0 1 0 0 INV

0 0 0 0 0 0 0 1 0 1 LSR

0 0 0 0 0 0 0 1 1 0 ROR

0 0 0 0 0 0 0 1 1 1 RRC

0 0 0 0 0 0 1 0 0 0 ASR

0 0 0 0 0 0 1 0 0 1 LSL

0 0 0 0 0 0 1 0 1 0 ROL

0 0 0 0 0 0 1 0 1 1 RLC

Branching Operations

$IR_{15\ ->8}$

1 1 1 1 1 0 0 0 BR

1 1 1 1 1 0 0 1 BEQ

1 1 1 1 1 0 1 0 BNE

1 1 1 1 1 0 1 1 BLO

1 1 1 1 1 1 0 0 BLS

1 1 1 1 1 1 0 1 BHI

1 1 1 1 1 1 1 0 BHS

No Operand Operations

$IR_{15\ ->0}$

1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 NOP

1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 HLT

Grouping

Next Address Field	G1 2 bits ALU	G2 3 bits	G3 3 bits
6 bits	0 0 Add 0 1 INC 1 0 DEC 1 1 Alu Activate	0 0 0 no action 0 0 1 PCout 0 1 0 MDRout 0 1 1 Zout 1 0 0 Rsrc-out 1 0 1 Rdst-out 1 1 0 Sourceout 1 1 1 IR-Addout	0 0 0 No Action 0 0 1 PCin 0 1 0 IRin 0 1 1 Zin 1 0 0 Rsrc-in 1 0 1 Rdst-in

G4 2 bits	G5 2 bits	G6 2 bits	G7 3 bits
0 0 No Action 0 1 MARin 1 0 MDRin	0 0 No Action 0 1 Yin 1 0 Source-in	0 0 No Action 0 1 READ 1 0 WRITE	0 0 0 No Action 0 0 1 Wide Branch 0 1 0 Branch_Check 0 1 1 Check in/direct Source 1 0 0 Check in/direct Dest 1 0 1 Dest_Branch 1 1 0 Check if dest register direct

Control store

000000	000001	PCout,MARin,RD,INC,Zin
000001	000010	Zout,PCin
000010	000011	MDRout,IRin
000011	Xxxxxxx	WideBranch
000100	000101/ 000000 (based on check)	Branch_check,Pcout,Yin
000110	000111	AddresFieldOfIR-out,ADD,Zin
000111	000000	Zout,PCin
001000	010110	RSout,Source-in(reg direct)//jump to dst branch
001001	010101	RSout,MARin,RD(reg in direct)
001010	001011	RSout,MARin,RD,INC,Zin(auto inc)
001011	010011	Zout,RSin//jump to check in/direct
001100	001101	RSout,DEC,Zin//(auto dec)
001101	010011	Zout,RSin,MARin,RD//jump to check in/direct
001110	001111	PCout,MARin,RD,INC,Zin(indexed)
001111	010000	Zout,PCin
010000	010001	MDRout,Yin
010001	010010	RSout,ADD,Zin
010010	010011	Zout,MARin,RD//jump to check in/direct

010011	010100	CheckDirect/InDirect(m7tageeeeeen)DDdd $[\text{Next add Field}]_0$ <- $[\text{Next add Field}]_0$ 'OR' $[IR]_9$
010100	010101	MDRout,MARin,RD
010101	010110	MDRout,Source-in(go branch for dst)
010110	01xxxx	BranchOnDist
010111	100100	RDSTout,MDRin(reg Direct)
011000	100100	RDStout,MARin,RD(reg in direct)
011001	011010	RDStout,MARin,RD,INC,Zin(Auto Inc)
011010	100010	RDStin,zout//jump to check direct/indirect
011011	011100	RDStout,DEC,Zin(auto dec)
011100	100010	Zout,MARin,RD,RDStin//jump to check direct/indirect
011101	011110	PCout,MARin,RD,INC,Zin(indexed)
011110	011111	Zout,PCin
011111	100000	MDRout,Yin
100000	100001	Rdstout,ADD,Zin
100001	100010	Zout,MARin,READ//jump to check direct/indirect
100010	100011	CheckDir/Indir DSt $[\text{Next add Field}]_{0,1,2}$ <- $[\text{Next add Field}]_{0,1,2}$ 'XOR' $[IR]_5$
100011	100100	MDRout,MARin,RD
100100	100101	Source-out,Yin
100101	100110	MDRout
100110	100111	ALU-activate,Zin
100111	101000	Check dst register direct or not $[\text{Next add Field}]_0$ <- $[[IR]_5.[IR]^c_3.[IR]^c_4]^c$
101000	000000	Zout,Rdstin
101001	000000	Zout,MDRin,WR
101010	000000	NOP

101011	101100	HLT
101100	101100	

Micro Instruction

Destination Addressing Modes

Register direct

Rdst-out

Register indirect

Rdst-out,MARin,READ

MDRout

Register autoincrement

Rdst-out,MARin,READ,INC,Zin

Zout,Rdst-in

“If direct”

MDRout

“If indirect”

MDRout,MARin,READ

MDRout

Register autodecrement

Rdst-out,DEC,Zin

Zout,MARin,Rdst-in,READ

“If direct”

MDRout

“If indirect”

MDRout,MARin,READ

MDRout

Register index

PCout,MARin,READ,INC,Zin

Zout,PCin

MDRout,Yin

Rdst-out,ADD,Zin

Zout,MARin,READ

“If direct”

MDRout

“If indirect”

MDRout,MARin,READ

MDRout

Source Addressing Modes

Register direct

Rsrc-out,Source-in

Register indirect

Rsrc-out,MARin,READ

MDRout,Source-in

Register autoincrement

Rsrc-out,MARin,READ,INC,Zin

Zout,Rsrc-in

“If direct”

MDRout,Source-in

“If indirect”

MDRout,MARin,READ

MDRout,Source-in

Register autodecrement

Rsrc-out,DEC,Zin

Zout,MARin,Rsrc-in,READ

“If direct”

MDRout,Source-in

“If indirect”

MDRout,MARin,READ

MDRout,Source-in

Register index

PCout,MARin,READ,INC,Zin

Zout,PCin

MDRout,Yin

Rsrc-out,ADD,Zin

Zout,MARin,READ

“If direct”

MDRout,Source-in

“If indirect”

MDRout,MARin,READ

MDRout,Source-in

Two Operands Micro Instructions

If Two Operand instruction:
ADD/ADC/SUB/SBC/OR/XNOR/AND

Source-out, ADD/ADC/SUB/SBC/OR/XNOR/AND,Zin

“if dest direct register”

Zout,Rdst-in

“If dest indirect register”

Zout,MDRin,WRITE

Case Two Operand instruction: MOV

“if dest direct register”

Source-out,Rdst-in

“if dest indirect register”

Source-out,MDRin,WRITE

Case Two Operand instruction :CMP

Source-out,CMP

One Operand Micro Instructions

If One Operand instruction Not Branching

One Operand Micro instruction,Zin

“if dest direct register”

Zout,Rdst-in

“if dest indirect register”

Zout,MDRin,WRITE

Branching

PCout,Yin, if ‘not branch’ then end

Address-field-of-IRout,ADD,Zin

Zout,PCin

CPI

Destination Addressing modes = [1 + 2 + (3+4) + (3+4) + (6+7)] = 30

Source Addressing modes = [1 + 2 + (3+4) + (3+4) + (6+7)] = 30

Branching = 3*7 = 21

Two Operands = 4*9 = 36

One Operands = 4*11 = 44

HLT and NOP = 2

Total CPI = 163

Number of memory access

We assumed that 2 operands have same addressing mode.

#memory access for each of OR / XNOR / ADD / ADC / SUB / SBC / MOV addressing modes:

Register mode	# memory access
Register indirect	1x2=2+1=3
Auto increment(direct)	1x2=2+1=3
Auto decrement(direct)	1x2=2+1=3
Index direct	2x2=4+1=5

Auto increment(indirect)	2x2=4+1=5
Auto decrement(indirect)	2x2=4+1=5
Index indirect	3x2=6+1=7

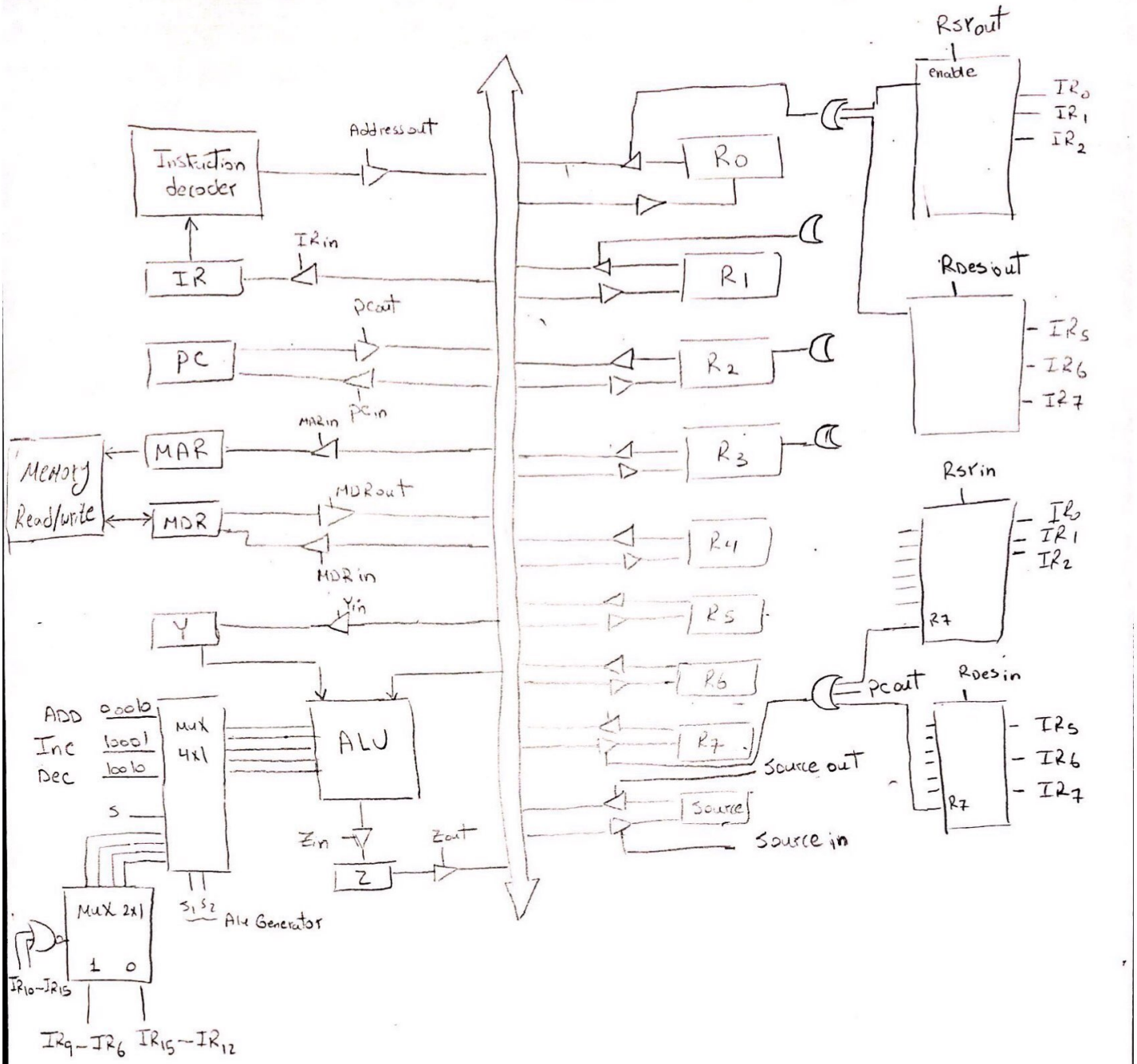
CMP instruction:

Register mode	# memory access
Register indirect	1x2=2
Auto increment(direct)	1x2=2
Auto decrement(direct)	1x2=2
Index direct	2x2=4
Auto increment(indirect)	2x2=4
Auto decrement(indirect)	2x2=4
Index indirect	3x2=6

One Operand Instruction:

Register mode	# memory access
Register indirect	1+1=2
Auto increment(direct)	1+1=2
Auto decrement(direct)	1+1=2
Index direct	2+1=3
Auto increment(indirect)	2+1=3
Auto decrement(indirect)	2+1=3
Index indirect	3+1=4

Scanned with CamScanner



Scanned with CamScanner