Lebanese American University

Department of Computer Science & Mathematics

CSC 320 — Computer Organization

A Simple RISC Processor

Design and Implementation

Walid Chebaro

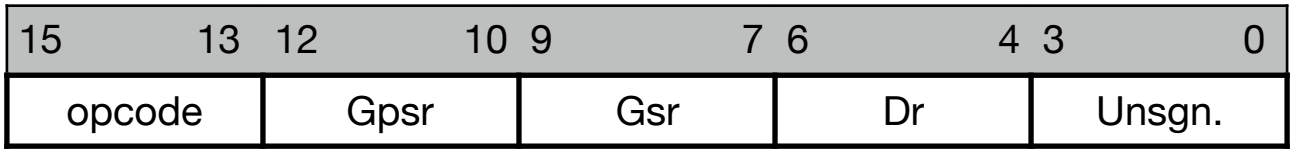
Ibrahim Meneim

Joseph Zakher

Submitted on 23/04/2019

Part I.

Q-Format



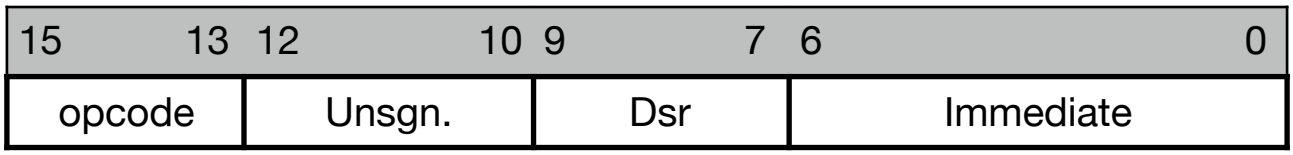
Description of the Q-Format

→ Takes an opcode (3 bits)

→ Takes three registers:
2 source registers Gsr and Gpsr (3 bits).
1 destination register Dr (3 bits).

→ Used mainly for arithmetic instructions and other instructions that requires 3 registers.

TR-Format



Description of the TR-Format

→ Takes an opcode (3 bits)

→ Takes one register:
Destination/source register Dsr (3 bits).

B-Format

Description of the B-Format

→ Takes an opcode (3 bits)

→ Takes two registers:
Dsr and Dr (3bits)

→ Takes a value or offset: Immediate (7 bits), shift left 5 gives 2^12 = 4096 bits (4Kb)

GOTO-Format

Description of the GOTO-Format

→ Takes an opcode (3 bits)

→ Takes an address (13 bits)

→ Used mainly to unconditionally jump to a given address.

Part II.

Core Instruction Set

NAME	MNEMONIC	FORMAT	OPERATION	OPCODE
Plus	plus	Q	Dr=Gsr+Gpsr	000
Minus	mnus	Q	Dr=Gsr-Gpsr	001
Read Word	rwrd	TR	Dsr=Imm	010
Write Word	wwrd	TR	Imm=Dsr	011
Read Immediate	rwid	TR	Dsr=Immediate	100
Branch Zero	bzer	B	if(Dsr-Dr=0) Branch to Imm.	101
Omega Branch	boxx	B	if(Dsr>Dr) Branch to Imm.	110
Go to	goto	GOTO	PC = Address	111

Part III.

Registers Identifiers

NAME	NUMBER	USE
$zero	ZERO	The Constant Value 0
$gpr0-$gpr7	0-7	General Purpose Reg.

Part IV.

Plus plus

→ Opcode 000
→ Q-Format.
→ Add the value of Gsr and Gpsr and store it in Dr.

Example

plus $gpr2, $gpr0, $gpr1 → $gpr2 = $gpr0 + $gpr1

000	$gpr0	$gpr1	$gpr2	0000

Minus mnus

→ Opcode 001
→ Q-Format.
→ Subtract the value of Gpsr from Gsr and store it in Dr.

Example

mns $gpr2, $gpr0, $gpr1 → $gpr2 = $gpr0 - $gpr1

001	$gpr0	$gpr1	$gpr2	0000

Read Word rwrd

→ Opcode 010
→ TR-Format.
→ Copy from memory to register.

Example

rwrd $gpr0, 8 → $gpr0 = 8

010	$gpr0	000	8

Write Word wwrd

→ Opcode 011
→ TR-Format.
→ Copy from register to memory.

Example

wwrd $gpr1, 8→ memory at address 8 = $gpr1

011	$gpr0	000	8

Read Immediate rwid

→ Opcode 100
→ TR-Format.
→ Loads the immediate value into register Dsr.

Example

rwi $gpr0, 3 → $gpr0 = 3

100	0	$gpr0	3

Branch Zero bzer

→ Opcode 101
→ TR-Format.
→ If the value of Dsr – Bar = 0 then branch to immediate.

Example

bzer $gpr1, $gpr0, loop → if ($gpr1 - $gpr0 = 0) then PC+2+Imm. 8

```
PC+2  …
PC+2  …
Loop: …
```

101	$gpr0	$gpr1	Loop

Omega Branch box

→ Opcode 110
→ TR-Format.
→ If the value of Dsr is greater than the value of Bar then branch to immediate.

Example

box $gpr1, $gpr0, loop → $gpr1 > $gpr0 then PC+4+Imm. 4
(Identical to ex. of Branch Zero)

110	$gpr0	$gpr1	Loop

Note that if we want to go to an address that takes more than 8 bits, we insert a Go To instruction a word after the Branch Zero or Omega Branch instruction.

Go To goto

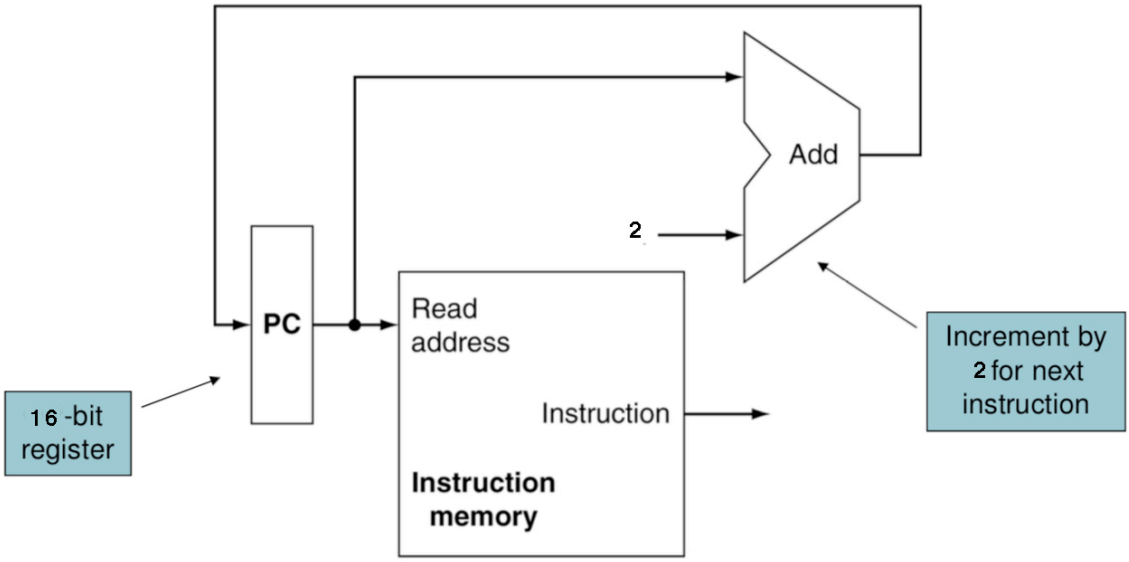
→ Opcode 111
→ GOTO-Format.
→ Go to target address.

Example

goto 1004 → PC = 1004

111	1004

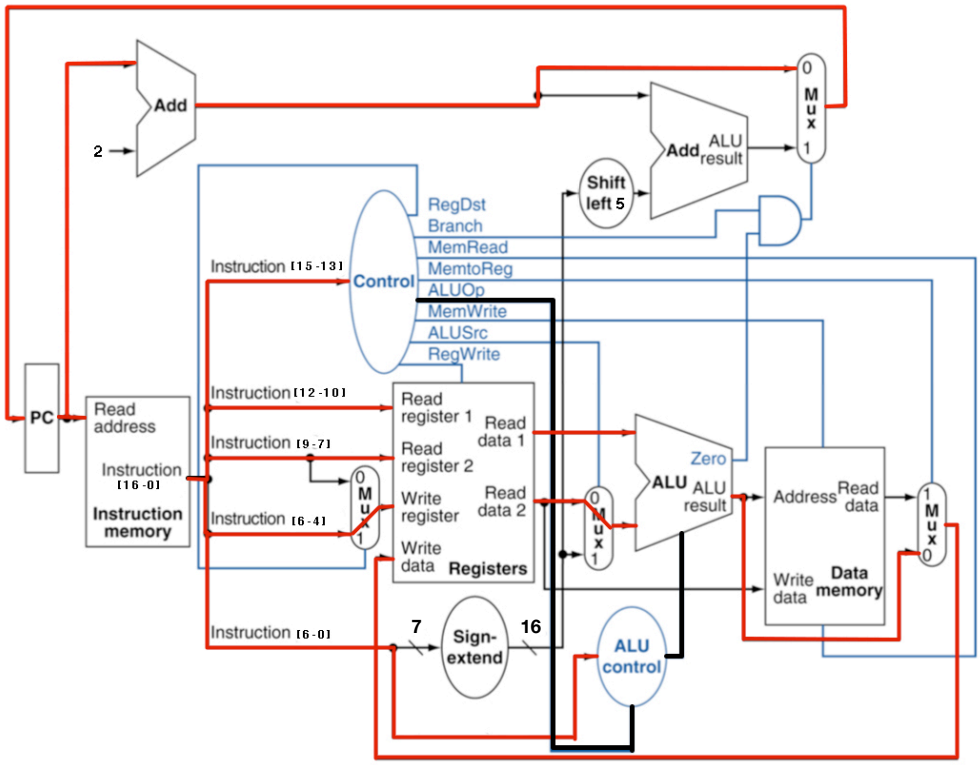
Part V.



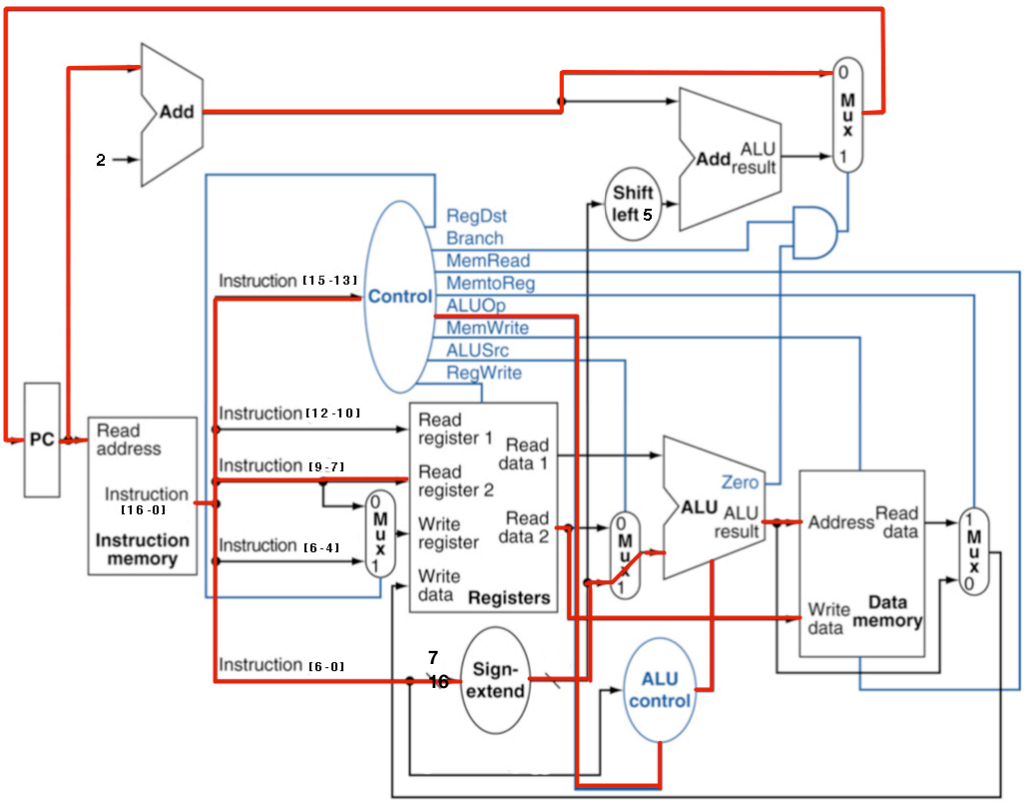
All the instructions are the same length: 16-bits.
Every cycle, therefore, **the PC** needs to be **incremented by 2** (16 bits = **2** bytes)

Part VI.

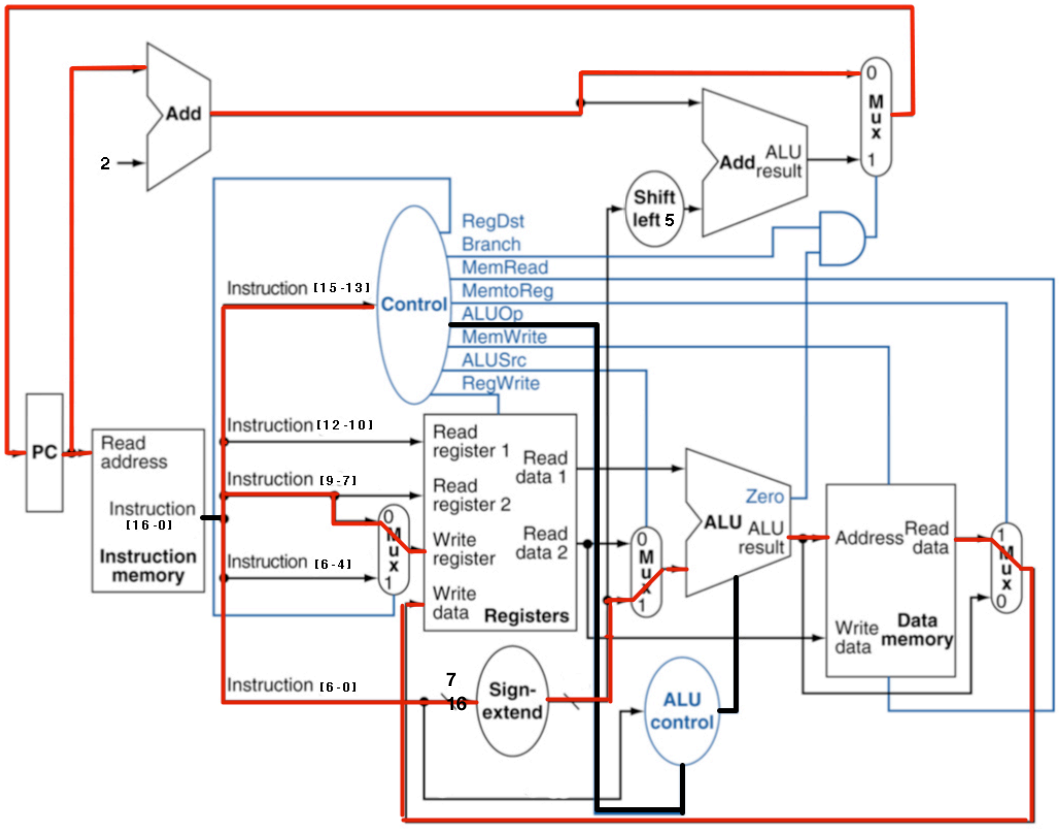
1. Datapath of Arithmetic Instructions



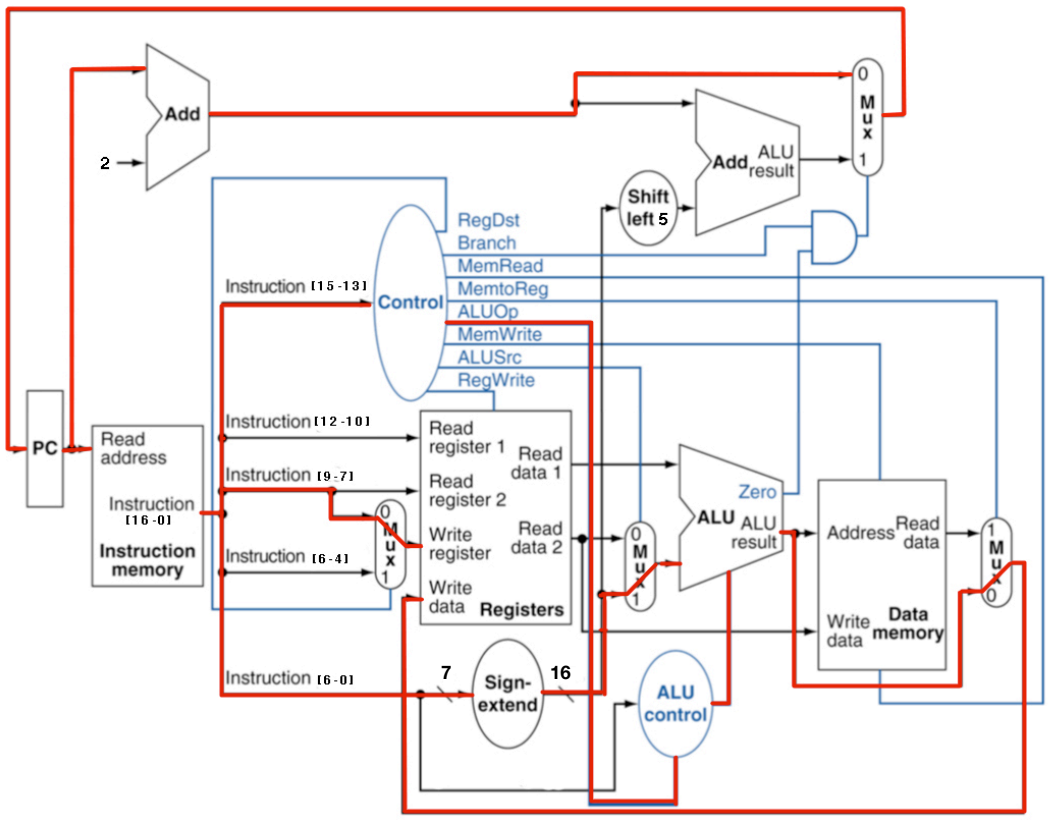
2. Datapath of Write Word Instruction



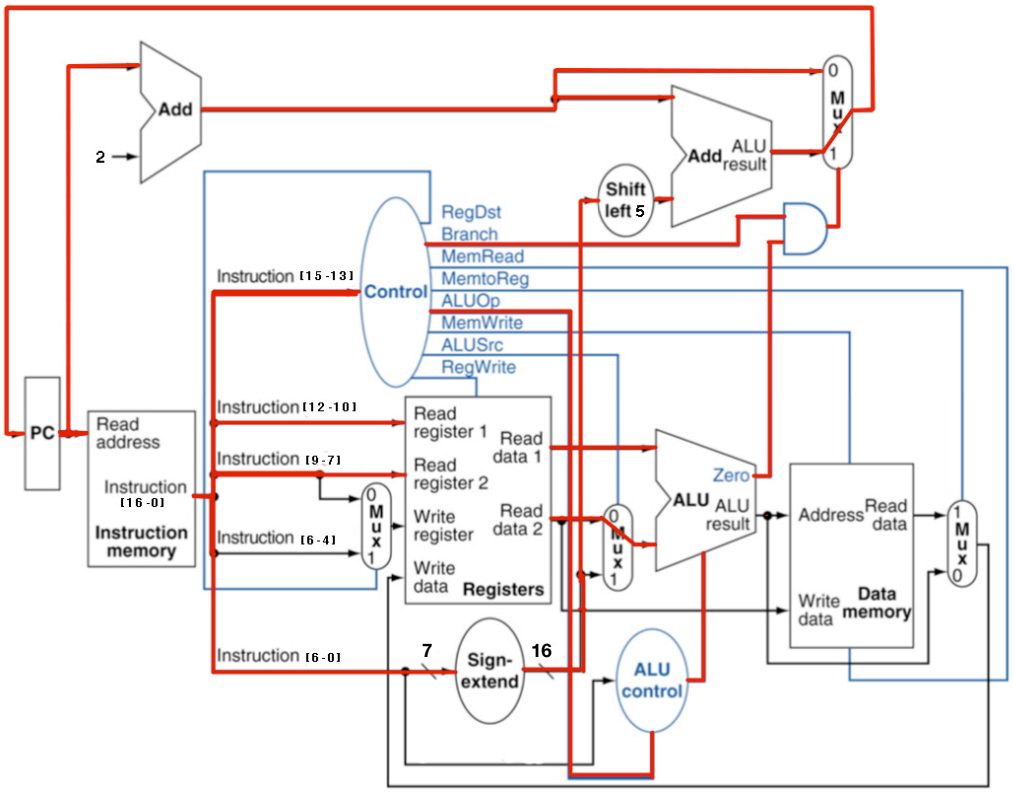
3. Datapath of Read Word Instruction



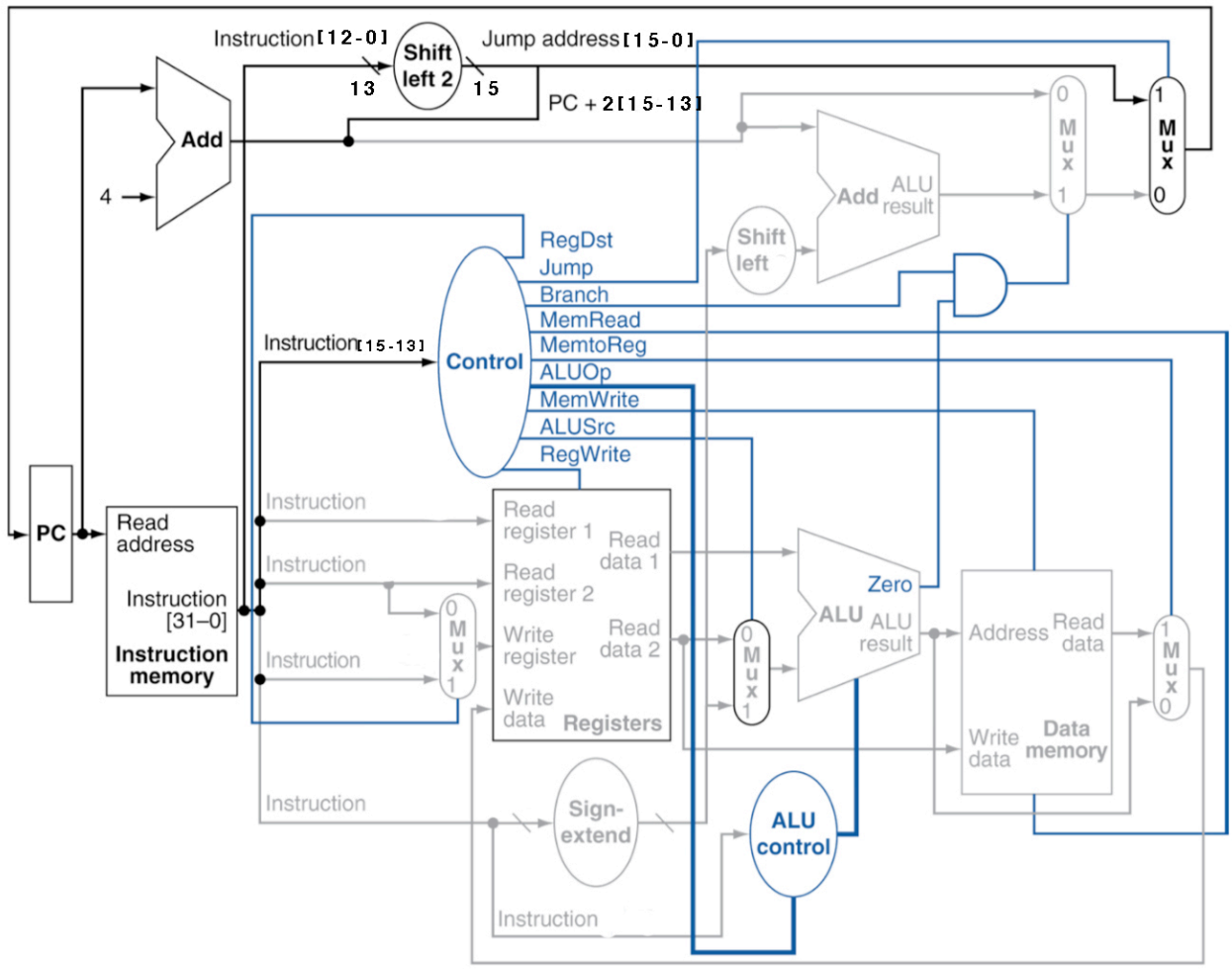
4. Datapath of Read Word Immediate Instruction



5. Datapath of Branch Zero / Omega Branch Instruction



6. Datapath of Go To Instruction



Part VII.

```java
import java.text.DecimalFormat;

public class Reg {

    int num;
    private int data;

    public Reg(int num, int data) {
        this.num = num;
        this.data = data;
    }

    public Reg() {

    }

    public int getNum() {
        return num;
    }

    public void setNum(int num) {
        this.num = num;
    }

    public int getData() {
        return data;
    }

    public void setData(int data) {
        this.data = data;
    }

    public Reg(int num) {
        this.num = num;
    }

    public String toStringNum() {

        switch (num) {
        case 0:
            return " 000 ";
        case 1:
            return " 001 ";
        case 2:
            return " 010 ";
        case 3:
            return " 011 ";
        case 4:
            return " 100 ";
        case 5:
            return " 101 ";
        case 6:
            return " 110 ";
        case 7:
            return " 111 ";
        }
        return null;

    }

    public String toStringData() {

        int a;
        int temp = data;
        String x = "";

        if (temp == 0)
            x = "0000";

        while (temp > 0) {
            a = temp % 2;
            x = a + "" + x;
            temp = temp / 2;
        }

        DecimalFormat f = new DecimalFormat("0000000");

        int y = Integer.parseInt(x);

        return f.format(y);
    }

}
```

```java
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Scanner;

//import java.util.Scanner;

public class Mem {

    String imm;

    // Store the instructions as MIPS
    String[] instructions = new String[7];

    // Memory size: 4Kb
    String[] address = new String[4095];

    // Store the instructions as machine language (binary)
    ArrayList<String> machineLanguage = new ArrayList<>();

    public Mem(String imm) {
        this.imm = imm;

    }

    public Mem() {

    }

    public void setInstructions(String[] instructions) {
        Scanner scan = new Scanner(System.in);
        for (int i = 0; i < instructions.length; i++) {
            instructions[i] = scan.nextLine();
        }
        scan.close();
    }

    public String Binary(int data) {

        int a;
        int temp = data;
        String x = "";

        if (temp == 0)
            x = "0000";

        while (temp > 0) {
            a = temp % 2;
            x = a + "" + x;
            temp = temp / 2;
        }

        DecimalFormat f = new DecimalFormat("0000000");

        int y = Integer.parseInt(x);

        return f.format(y);
    }

    public String BinaryJ(int data) {

        int a;
        int temp = data;
        String x = "";

        if (temp == 0)
            x = "0000";

        while (temp > 0) {
            a = temp % 2;
            x = a + "" + x;
            temp = temp / 2;
        }

        DecimalFormat f = new DecimalFormat("0000000000000");

        int y = Integer.parseInt(x);

        return f.format(y);
    }

    @Override
    public String toString() {
        return "Mem [machineLanguage=" + machineLanguage + "]";
    }

}
```

```java
import java.util.Scanner;

public class ALU {

    Scanner scan = new Scanner(System.in);

    public static void arithmetic(Reg gpr, Reg gpr1, Reg gpr2, String op, Mem memory) {

        if (op.equals("plus")) {
            gpr.setData(gpr1.getData() + gpr2.getData());
            System.out.println("$gprs" + gpr.num + " data: " + gpr.getData());
//          System.out.println("opcode: 000 ");
            memory.machineLanguage
                    .add("000 " + gpr.toStringNum() + gpr1.toStringNum() + gpr2.toStringNum() + " 0000");
        } else if (op.equals("mnus")) {
            gpr.setData(gpr1.getData() - gpr2.getData());
            System.out.println(gpr.getData());
//          System.out.println("opcode: 001 ");
            memory.machineLanguage
                    .add("001" + gpr.toStringNum() + " " + gpr1.toStringNum() + " " + gpr2.toStringNum() + " 000");

        }

    }

}
```

```java
public class pc {

    int inc = 0;

    public pc(int inc) {
        this.inc = inc;
        // TODO Auto-generated constructor stub

    }

    public pc() {

    }
```

```java
public class MUX {

    int input1;
    int input2;
    String output;

    int signal;

    public MUX(int signal) {
        this.signal = signal;

    }

    int mux(int signal, String input1, String input2) {
        if (signal == 1)
            output = input2;
        else if (signal == 0)
            output = input1;
        else
            System.out.println("wrong signal type, please insert a 1 or a 0.");

        return signal;
    }

    public MUX() {

    }

}
```

```java
import java.util.Scanner;

public class controlUnit {

    Scanner scan = new Scanner(System.in);

    void select(Mem memory, dp datapath) {

        pc pc = new pc();
        int x = 0;
        int y = 0;
        int i;

        for (i = 0; i < memory.instructions.length; i++) {

            if (memory.instructions[i].substring(0, 4).equals("plus")
                    || memory.instructions[i].substring(0, 4).equals("mnus")) {

                datapath.regDst.mux(0, memory.instructions[i].substring(14, 19),
                        memory.instructions[i].substring(22, 27));

                datapath.aluSrc.mux(0, memory.instructions[i].substring(14, 19),
                        memory.instructions[i].substring(22, 27));

                datapath.memToReg.mux(0, memory.instructions[i].substring(6, 11), null);

                datapath.branch.mux(0, null, null);

                switch (memory.instructions[i].substring(6)) {
                case "$gprs0, $gprs0, $gprs0":
                    ALU.arithmetic(datapath.$gprs0, datapath.$gprs0, datapath.$gprs0,
                            memory.instructions[i].substring(0, 4), memory);
                    break;
                case "$gprs0, $gprs0, $gprs1":
                    ALU.arithmetic(datapath.$gprs0, datapath.$gprs0, datapath.$gprs1,
                            memory.instructions[i].substring(0, 4), memory);
                    break;
                case "$gprs0, $gprs1, $gprs0":
                    ALU.arithmetic(datapath.$gprs0, datapath.$gprs1, datapath.$gprs0,
                            memory.instructions[i].substring(0, 4), memory);
                    break;
                case "$gprs0, $gprs1, $gprs1":
                    ALU.arithmetic(datapath.$gprs0, datapath.$gprs1, datapath.$gprs1,
                            memory.instructions[i].substring(0, 4), memory);
                    break;
                case "$gprs1, $gprs0, $gprs0":
                    ALU.arithmetic(datapath.$gprs1, datapath.$gprs0, datapath.$gprs0,
                            memory.instructions[i].substring(0, 4), memory);
                    break;
                case "$gprs1, $gprs0, $gprs1":
                    ALU.arithmetic(datapath.$gprs1, datapath.$gprs0, datapath.$gprs1,
                            memory.instructions[i].substring(0, 4), memory);
                    break;
                case "$gprs1, $gprs1, $gprs0":
                    ALU.arithmetic(datapath.$gprs1, datapath.$gprs1, datapath.$gprs0,
                            memory.instructions[i].substring(0, 4), memory);
                    break;
                case "$gprs1, $gprs1, $gprs1":
                    ALU.arithmetic(datapath.$gprs1, datapath.$gprs1, datapath.$gprs1,
                            memory.instructions[i].substring(0, 4), memory);
                    break;
                case "$gprs0, $gprs1, $gprs2":
                    ALU.arithmetic(datapath.$gprs0, datapath.$gprs1, datapath.$gprs2,
                            memory.instructions[i].substring(0, 4), memory);
                    break;
                case "$gprs2, $gprs2, $gprs4":
                    ALU.arithmetic(datapath.$gprs2, datapath.$gprs2, datapath.$gprs4,
                            memory.instructions[i].substring(0, 4), memory);
                    break;
                case "$gprs1, $gprs1, $gprs3":
                    ALU.arithmetic(datapath.$gprs1, datapath.$gprs1, datapath.$gprs3,
                            memory.instructions[i].substring(0, 4), memory);
                    break;

                // CONTINUE FOR ALL REGISTERS
                // (343 possibilities!)

                }

            }

            else if (memory.instructions[i].substring(0, 4).equals("rwrd")) {
                memory.imm = memory.instructions[i].substring(14);

                datapath.aluSrc.mux(1, memory.instructions[i].substring(14), null);

                datapath.memToReg.mux(1, null, null);

                datapath.branch.mux(0, null, null);
```

```
88
89              datapath.branch.mux(0, null, null);
90
91              switch (memory.instructions[i].charAt(11)) {
92              case '0':
93                  datapath.$gprs0.setData(Integer.parseInt(memory.imm));
94                  memory.machineLanguage
95                          .add("010 000" + datapath.$gprs0.toStringNum() + datapath.$gprs0.toStringData());
96                  break;
97              case '1':
98                  datapath.$gprs1.setData(Integer.parseInt(memory.imm));
99                  memory.machineLanguage
100                         .add("010 000" + datapath.$gprs1.toStringNum() + datapath.$gprs1.toStringData());
101                 break;
102             case '2':
103                 datapath.$gprs2.setData(Integer.parseInt(memory.imm));
104                 memory.machineLanguage
105                         .add("010 000" + datapath.$gprs2.toStringNum() + datapath.$gprs2.toStringData());
106                 break;
107             case '3':
108                 datapath.$gprs3.setData(Integer.parseInt(memory.imm));
109                 memory.machineLanguage
110                         .add("010 000" + datapath.$gprs3.toStringNum() + datapath.$gprs3.toStringData());
111                 break;
112             case '4':
113                 datapath.$gprs4.setData(Integer.parseInt(memory.imm));
114                 memory.machineLanguage
115                         .add("010 000" + datapath.$gprs4.toStringNum() + datapath.$gprs4.toStringData());
116                 break;
117             case '5':
118                 datapath.$gprs5.setData(Integer.parseInt(memory.imm));
119                 memory.machineLanguage
120                         .add("010 000" + datapath.$gprs5.toStringNum() + datapath.$gprs5.toStringData());
121                 break;
122             case '6':
123                 datapath.$gprs6.setData(Integer.parseInt(memory.imm));
124                 memory.machineLanguage
125                         .add("010 000" + datapath.$gprs6.toStringNum() + datapath.$gprs6.toStringData());
126                 break;
127             case '7':
128                 datapath.$gprs7.setData(Integer.parseInt(memory.imm));
129                 memory.machineLanguage
130                         .add("010 000" + datapath.$gprs7.toStringNum() + datapath.$gprs7.toStringData());
131                 break;
132             }
133         }
134
135         else if (memory.instructions[i].substring(0, 4).equals("wwrd")) {
136
137             datapath.regDst.mux(1, memory.instructions[i].substring(6, 11), null);
138
139             datapath.aluSrc.mux(1, memory.instructions[i].substring(14), null);
140
141             datapath.branch.mux(0, null, null);
142
143             switch (memory.instructions[i].charAt(11)) {
144             case '0':
145                 memory.imm = String.valueOf(datapath.$gprs0.getData());
146                 memory.machineLanguage
147                         .add("011 000" + datapath.$gprs0.toStringNum() + datapath.$gprs0.toStringData());
148                 break;
149             case '1':
150                 memory.imm = String.valueOf(datapath.$gprs1.getData());
151                 memory.machineLanguage
152                         .add("011 000" + datapath.$gprs1.toStringNum() + datapath.$gprs1.toStringData());
153                 break;
154             case '2':
155                 memory.imm = String.valueOf(datapath.$gprs2.getData());
156                 memory.machineLanguage
157                         .add("011 000" + datapath.$gprs2.toStringNum() + datapath.$gprs2.toStringData());
158                 break;
159             case '3':
160                 memory.imm = String.valueOf(datapath.$gprs3.getData());
161                 memory.machineLanguage
162                         .add("011 000" + datapath.$gprs3.toStringNum() + datapath.$gprs3.toStringData());
163                 break;
164             case '4':
165                 memory.imm = String.valueOf(datapath.$gprs4.getData());
166                 memory.machineLanguage
167                         .add("011 000" + datapath.$gprs4.toStringNum() + datapath.$gprs4.toStringData());
168                 break;
169             case '5':
170                 memory.imm = String.valueOf(datapath.$gprs5.getData());
171                 memory.machineLanguage
172                         .add("011 000" + datapath.$gprs5.toStringNum() + datapath.$gprs5.toStringData());
173                 break;
174             case '6':
175                 memory.imm = String.valueOf(datapath.$gprs6.getData());
176                 memory.machineLanguage
177                         .add("011 000" + datapath.$gprs6.toStringNum() + datapath.$gprs6.toStringData());
```

```
                case '7':
                    memory.imm = String.valueOf(datapath.$gprs7.getData());
                    memory.machineLanguage
                            .add("011 000" + datapath.$gprs7.toStringNum() + datapath.$gprs7.toStringData());
                    break;
                }

                datapath.memToReg.mux(1, memory.imm, null);

                memory.address[Integer.parseInt(memory.instructions[i].substring(14))] = memory.imm;

            }

            else if (memory.instructions[i].substring(0, 4).equals("rwid")) {

                datapath.regDst.mux(0, memory.instructions[i].substring(6, 11), null);

                datapath.aluSrc.mux(1, memory.instructions[i].substring(14), null);

                datapath.branch.mux(0, null, null);

                datapath.memToReg.mux(0, memory.instructions[i].substring(14), null);

                switch (memory.instructions[i].charAt(11)) {
                case '0':
                    datapath.$gprs0.setData(Integer.parseInt(memory.instructions[i].substring(14)));
                    System.out.println("$gprs0 data set to " + datapath.$gprs0.getData());
                    memory.machineLanguage
                            .add("100 000" + datapath.$gprs0.toStringNum() + datapath.$gprs0.toStringData());
                    break;
                case '1':
                    datapath.$gprs1.setData(Integer.parseInt(memory.instructions[i].substring(14)));
                    System.out.println("$gprs1 data set to " + memory.instructions[i].substring(14));
                    memory.machineLanguage
                            .add("100 000" + datapath.$gprs1.toStringNum() + datapath.$gprs1.toStringData());
                    break;
                case '2':
                    datapath.$gprs2.setData(Integer.parseInt(memory.instructions[i].substring(14)));
                    System.out.println("$gprs2 data set to " + memory.instructions[i].substring(14));
                    memory.machineLanguage
                            .add("100 000" + datapath.$gprs2.toStringNum() + datapath.$gprs2.toStringData());
                    break;
                case '3':
                    datapath.$gprs3.setData(Integer.parseInt(memory.instructions[i].substring(14)));
                    System.out.println("$gprs3 data set to " + memory.instructions[i].substring(14));
                    memory.machineLanguage
                            .add("100 000" + datapath.$gprs3.toStringNum() + datapath.$gprs3.toStringData());
                    break;
                case '4':
                    datapath.$gprs4.setData(Integer.parseInt(memory.instructions[i].substring(14)));
                    System.out.println("$gprs4 data set to " + memory.instructions[i].substring(14));
                    memory.machineLanguage
                            .add("100 000" + datapath.$gprs4.toStringNum() + datapath.$gprs4.toStringData());
                    break;
                case '5':
                    datapath.$gprs5.setData(Integer.parseInt(memory.instructions[i].substring(14)));
                    System.out.println("$gprs5 data set to " + memory.instructions[i].substring(14));
                    memory.machineLanguage
                            .add("100 000" + datapath.$gprs5.toStringNum() + datapath.$gprs5.toStringData());
                    break;
                case '6':
                    datapath.$gprs6.setData(Integer.parseInt(memory.instructions[i].substring(14)));
                    System.out.println("$gprs6 data set to " + memory.instructions[i].substring(14));
                    memory.machineLanguage
                            .add("100 000" + datapath.$gprs6.toStringNum() + datapath.$gprs6.toStringData());
                    break;
                case '7':
                    datapath.$gprs7.setData(Integer.parseInt(memory.instructions[i].substring(14)));
                    System.out.println("$gprs7 data set to " + memory.instructions[i].substring(14));
                    memory.machineLanguage
                            .add("100 000" + datapath.$gprs7.toStringNum() + datapath.$gprs7.toStringData());
                    break;
                }

            }

            else if (memory.instructions[i].substring(0, 4).equals("bzer")) {

                datapath.regDst.mux(0, null, null);

                datapath.aluSrc.mux(0, memory.instructions[i].substring(14, 19), null);

                datapath.memToReg.mux(0, null, null);

                datapath.branch.mux(1, memory.instructions[i].substring(22), null);

                int j = 0;
                int k = 0;

                switch (memory.instructions[i].charAt(11)) {
```

```
            switch (memory.instructions[i].charAt(11)) {
            case '0':
                x = datapath.$gprs0.getData();
                j = 0;
                break;
            case '1':
                x = datapath.$gprs1.getData();
                j = 1;
                break;
            case '2':
                x = datapath.$gprs2.getData();
                j = 2;
                break;
            case '3':
                x = datapath.$gprs3.getData();
                j = 3;
                break;
            case '4':
                x = datapath.$gprs4.getData();
                j = 4;
                break;
            case '5':
                x = datapath.$gprs5.getData();
                j = 5;
                break;
            case '6':
                x = datapath.$gprs6.getData();
                j = 6;
                break;
            case '7':
                x = datapath.$gprs7.getData();
                j = 7;
                break;
            }

            switch (memory.instructions[i].charAt(19)) {
            case '0':
                y = datapath.$gprs0.getData();
                k = 0;
                break;
            case '1':
                y = datapath.$gprs1.getData();
                k = 1;
                break;
            case '2':
                y = datapath.$gprs2.getData();
                k = 2;
                break;
            case '3':
                y = datapath.$gprs3.getData();
                k = 3;
                break;
            case '4':
                y = datapath.$gprs4.getData();
                k = 4;
                break;
            case '5':
                y = datapath.$gprs5.getData();
                k = 5;
                break;
            case '6':
                y = datapath.$gprs6.getData();
                k = 6;
                break;
            case '7':
                y = datapath.$gprs7.getData();
                k = 7;
                break;
            }

            datapath.gpr.setNum(j);
            datapath.gprs.setNum(k);

            int b = Integer.parseInt(memory.instructions[i].substring(22));

            if (x - y == 0) {
                pc.inc = Integer.parseInt(memory.instructions[i].substring(22));
                System.out.println("branched to adress: " + memory.instructions[i].substring(22));
                System.out.println(memory.instructions[pc.inc]);
                memory.machineLanguage
                        .add("101 " + datapath.gpr.toStringNum() + datapath.gprs.toStringNum() + memory.Binary(b));
                break;
            } else {
                System.out.println(x + " is not equal to " + y + ".");
                memory.machineLanguage
                        .add("101 " + datapath.gpr.toStringNum() + datapath.gprs.toStringNum() + memory.Binary(b));

            }
        }

```

```
358             else if (memory.instructions[i].substring(0, 4).equals("boxx")) {
359
360                 datapath.regDst.mux(0, null, null);
361
362                 datapath.aluSrc.mux(0, memory.instructions[i].substring(14, 19), null);
363
364                 datapath.memToReg.mux(0, null, null);
365
366                 datapath.branch.mux(1, memory.instructions[i].substring(22), null);
367
368                 int j = 0;
369                 int k = 0;
370
371                 switch (memory.instructions[i].charAt(11)) {
372                 case '0':
373                     x = datapath.$gprs0.getData();
374                     j = 0;
375                     break;
376                 case '1':
377                     x = datapath.$gprs1.getData();
378                     j = 1;
379                     break;
380                 case '2':
381                     x = datapath.$gprs2.getData();
382                     j = 2;
383                     break;
384                 case '3':
385                     x = datapath.$gprs3.getData();
386                     j = 3;
387                     break;
388                 case '4':
389                     x = datapath.$gprs4.getData();
390                     j = 4;
391                     break;
392                 case '5':
393                     x = datapath.$gprs5.getData();
394                     j = 5;
395                     break;
396                 case '6':
397                     x = datapath.$gprs6.getData();
398                     j = 6;
399                     break;
400                 case '7':
401                     x = datapath.$gprs7.getData();
402                     j = 7;
403                     break;
404                 }
405
406                 switch (memory.instructions[i].charAt(19)) {
407                 case '0':
408                     y = datapath.$gprs0.getData();
409                     k = 0;
410                     break;
411                 case '1':
412                     y = datapath.$gprs1.getData();
413                     k = 1;
414                     break;
415                 case '2':
416                     y = datapath.$gprs2.getData();
417                     k = 2;
418                     break;
419                 case '3':
420                     y = datapath.$gprs3.getData();
421                     k = 3;
422                     break;
423                 case '4':
424                     y = datapath.$gprs4.getData();
425                     k = 4;
426                     break;
427                 case '5':
428                     y = datapath.$gprs5.getData();
429                     k = 5;
430                     break;
431                 case '6':
432                     y = datapath.$gprs6.getData();
433                     k = 6;
434                     break;
435                 case '7':
436                     y = datapath.$gprs7.getData();
437                     k = 7;
438                     break;
439                 }
440
441                 datapath.gpr.setNum(j);
442                 datapath.gprs.setNum(k);
443
444                 int b = Integer.parseInt(memory.instructions[i].substring(22));
445
446                 if (x > y) {
```

```
445
446                 if (x > y) {
447                     pc.inc = Integer.parseInt(memory.instructions[i].substring(22));
448                     System.out.println("branched to adress: " + memory.instructions[i].substring(22));
449                     System.out.println(memory.instructions[pc.inc]);
450                     memory.machineLanguage
451                             .add("110 " + datapath.gpr.toStringNum() + datapath.gprs.toStringNum() + memory.Binary(b));
452                     break;
453                 } else {
454                     System.out.println(x + " is not greater than " + y + ".");
455                     memory.machineLanguage
456                             .add("110 " + datapath.gpr.toStringNum() + datapath.gprs.toStringNum() + memory.Binary(b));
457                 }
458             }
459
460             else if (memory.instructions[i].substring(0, 4).equals("goto")) {
461                 pc.inc = Integer.parseInt(memory.instructions[i].substring(6));
462                 i = pc.inc - 1;
463                 memory.machineLanguage.add("111 " + memory.BinaryJ(pc.inc));
464             } else
465                 System.out.println(memory.instructions[i]);
466         }
467     }
```

```
1  import java.util.Iterator;
2  import java.util.Scanner;
3
4  public class processor {
5
6      controlUnit cu = new controlUnit();
7      dp datapath = new dp();
8
9      void process(Mem memory, dp datapath) {
10         cu.select(memory, datapath);
11     }
12
13     public static void main(String[] args) {
14
15         Scanner scan = new Scanner(System.in);
16         Mem memory = new Mem();
17         processor p = new processor();
18
19         // C Code Example
20         int m = 0;
21         int y = 5;
22
23         for (int i = 0; i < 10; i++) {
24             m += y;
25         }
26
27         // Setting $gprs2 to m, $gprs3 to 1 (i++) and $gprs4 to y.
28         p.datapath.$gprs2.setData(0);
29         p.datapath.$gprs3.setData(1);
30         p.datapath.$gprs4.setData(y);
31
32         // C Code Converted to MIPS.
33         memory.instructions[0] = "rwid  $gprs0, 10";
34         memory.instructions[1] = "rwid  $gprs1, 0";
35         memory.instructions[2] = "bzer  $gprs1, $gprs0, 6";
36         memory.instructions[3] = "plus  $gprs2, $gprs2, $gprs4";
37         memory.instructions[4] = "plus  $gprs1, $gprs1, $gprs3";
38         memory.instructions[5] = "goto  2";
39         memory.instructions[6] = "end:";
40
41         // MIPS Converted to Machine Language.
42         p.process(memory, p.datapath);
43
44         // Saving the Instructions in the Memory.
45         Iterator<String> it = memory.machineLanguage.iterator();
46         int i = 0;
47         while (it.hasNext()) {
48             memory.address[i] = it.next();
49             i += 2;
50         }
51
52         // Printing the Address Only when Filled with Instructions.
53         for (int j = 0; j < memory.address.length; j++) {
54             if (memory.address[j] != null)
55                 System.out.println(j + "    " + memory.address[j]);
56         }
57
58         scan.close();
59     }
60
61 }
62
```

```java

public class dp {

    // Registers
    Reg gpr = new Reg();
    Reg gprs = new Reg();

    Reg $gprs0 = new Reg(0);
    Reg $gprs1 = new Reg(1);
    Reg $gprs2 = new Reg(2);
    Reg $gprs3 = new Reg(3);
    Reg $gprs4 = new Reg(4);
    Reg $gprs5 = new Reg(5);
    Reg $gprs6 = new Reg(6);
    Reg $gprs7 = new Reg(7);

    // ALU
    ALU alu = new ALU();

    // Multiplexers
    MUX regDst = new MUX();
    MUX aluSrc = new MUX();
    MUX memToReg = new MUX();
    MUX branch = new MUX();

    public dp(Reg $gprs0, Reg $gprs1, Reg $gprs2, Reg $gprs3, Reg $gprs4, Reg $gprs5, Reg $gprs6, Reg $gprs7) {
        this.$gprs0 = $gprs0;
        this.$gprs1 = $gprs1;
        this.$gprs2 = $gprs2;
        this.$gprs3 = $gprs3;
        this.$gprs4 = $gprs4;
        this.$gprs5 = $gprs5;
        this.$gprs6 = $gprs6;
        this.$gprs7 = $gprs7;
    }

    public dp() {
    }

}

```

```
int k = 0;
int y = 1;
for (int i = 0; i < 10; i++) {
k += y;
}
```

Convert to Assembly languange

```
rwid  $gprs0, 10
rwid  $gprs1, 0
bzer  $gprs1, $gprs0, 6
plus  $gprs2, $gprs2, $gprs4
plus  $gprs1, $gprs1, $gprs3
goto  2
end:
```

```
rwid  $gprs0, 10
rwid  $gprs1, 0
bzer  $gprs1, $gprs0, 6
plus  $gprs2, $gprs2, $gprs4
plus  $gprs1, $gprs1, $gprs3
goto  2
end:
```

Convert to Machine languange

```
address 0: 100 000 000 0001010
address 2: 100 000 001 0000000
address 4: 101  001  000 0000110
address 6: 000  010  010  100  0000
address 8: 000  001  001  011  0000
address 10: 111 0000000000010
address 12: 101  001  000 0000110
address 14: 000  010  010  100  0000
address 16: 000  001  001  011  0000
address 18: 111 0000000000010
address 20: 101  001  000 0000110
address 22: 000  010  010  100  0000
address 24: 000  001  001  011  0000
address 26: 111 0000000000010
address 28: 101  001  000 0000110
address 30: 000  010  010  100  0000
address 32: 000  001  001  011  0000
address 34: 111 0000000000010
```

```
address 84: 101  001  000 0000110
```

Set DataPath Instructions