Title

Author

9 September 2017

1 Introduction

This document gives a brief overview of a 7-stage 2-issues pipelined RISC processor.

2 Pipeline Stages

2.1 Fetch stage

The fetch stage is the stage responsible for accessing the instruction memory. Note that two memories are used: one for instructions, and the other for data. And since this is a two-issue processor, we fetch two instructions instead of one. The fetch buffer consists of:

1. `NOP1:` A one-bit flag to indicate if the first issue is a NOP.
2. `IR1:` The instruction word of the first issue.
3. `NOP2:` A one-bit flag to indicate if the second issue is a NOP.
4. `IR2:` The instruction word of the second issue.

2.1.1 PC generator

The PC generator is a combinational circuit responsible for updating the PC every clock-cycle. See Figure 1 for a schematic of the component. There are four cases:

1. $PC \leftarrow [PC] + 2$. This is the normal case.
2. $PC \leftarrow address$. Needed with jumps and function calls.
3. $PC \leftarrow [PC] - dec$. This is used with incompatible issues.
4. $PC \leftarrow [PC]$. The PC needs to holds its value if the pipe is being stalled.

We define the functionality of the generator with the following priority:

```
if      (jump)          pcgen_out = addr
else if (hold)          pcgen_out = PC
else if (dec != 0)      pcgen_out = PC - dec
else                    pcgen_out = PC + 2
```

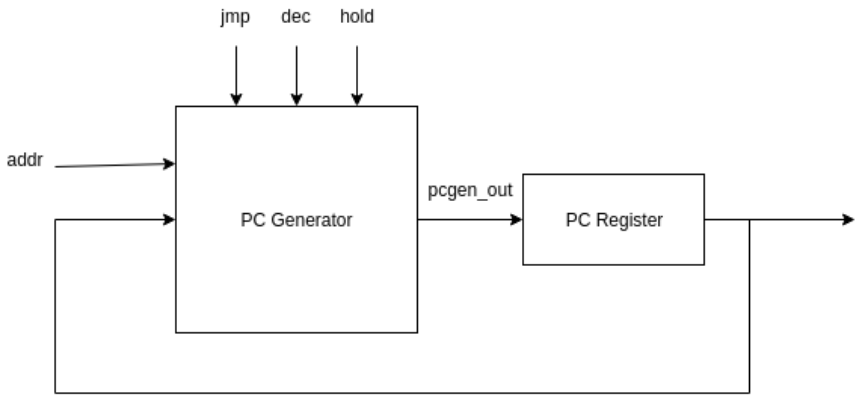

Figure 1: A schematic of the PC generator.

2.2 Pre-decode stage

The Pre-decode stage is responsible for:

1. Checking that two issues are compatible.
2. Breaking complex instructions into simpler ones.

The `IPD/ID` buffer consists of:

1. `NOP1:` A one-bit flag to indicate if the first issue is a NOP.
2. `IR1:` The instruction word of the first issue.
3. `NOP2:` A one-bit flag to indicate if the second issue is a NOP.
4. `IR2:` The instruction word of the second issue.

2.2.1 Non-compatible Issues

Any two issues are not compatible if one of the following conditions holds:

1. One of the issues is a `push` or `pop` instruction.
2. One of the issues is a `call` instruction.
3. The first issue is a branch or change of control operation.
4. The first issue loads a value, and the second uses it.
5. The second issue is `LDM`.
6. Both of the issues use the memory.

If two issues are not compatible, a signal is sent to the PC generator to decrement by 3, the second issue becomes NOP, and the previous stage is flushed. The Pre-decode buffer structure:

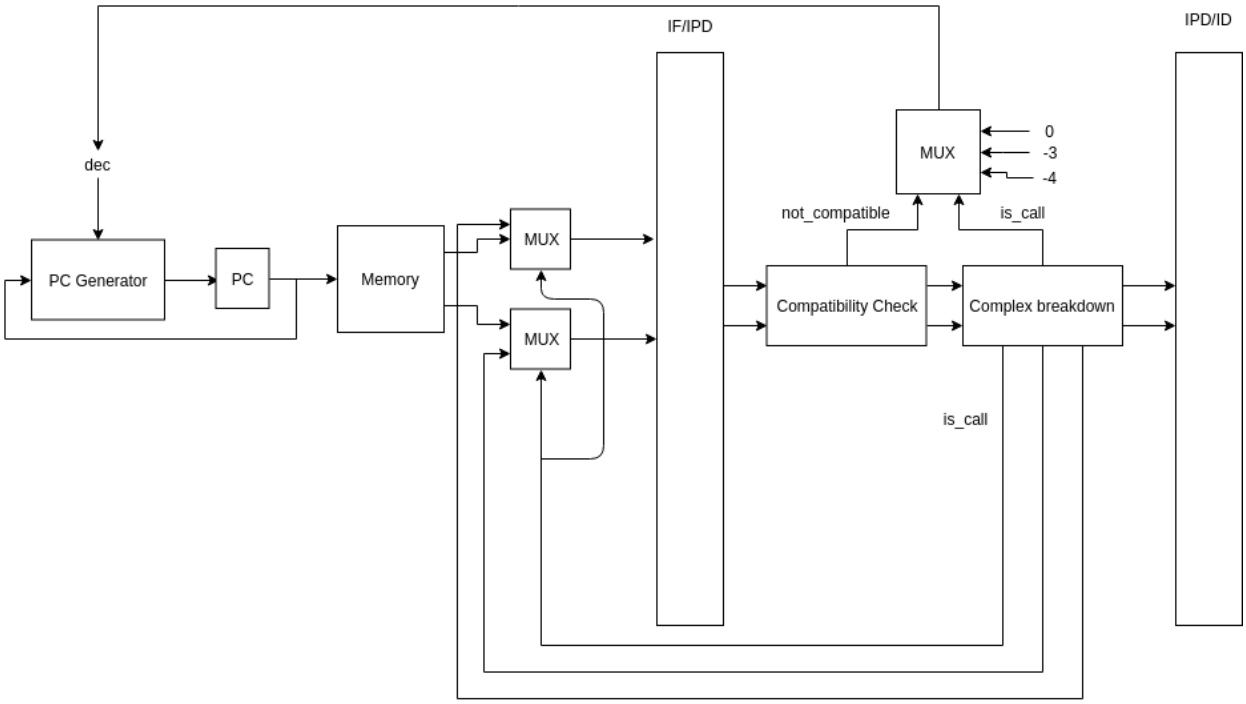

Figure 2: A schematic of the pre-decode stage.

2.2.2 Complex Instructions

The following instructions are considered complex:

1. *PUSH Rdst:* This does two operations: it stores `Rdst` in `@SP` and decrements `SP`. Hence it's broken into:

   ```
   STD     Rdst , SP
   DEC     SP
   ```

2. *POP Rdst:* Similar to `PUSH`. It's broken into

   ```
   INC     SP
   LDD     Rdst , SP
   ```

3. *CALL Rdst:* This one requires special treatment, since it gets broken into 3 instructions, which needs two packets:

   ```
   STD     PC,     SP
   DEC     SP
   MOV     PC,     Rdst
   ```

   In this case, the `PC` is decremented by 4, and it writes to the `IF/IPD` buffer the third instruction `MOV PC, Rdst` instead of reading it from memory.

2.3 Decode stage

See Figure 3 for a schematic of the stage. The Decode stage is responsible for:

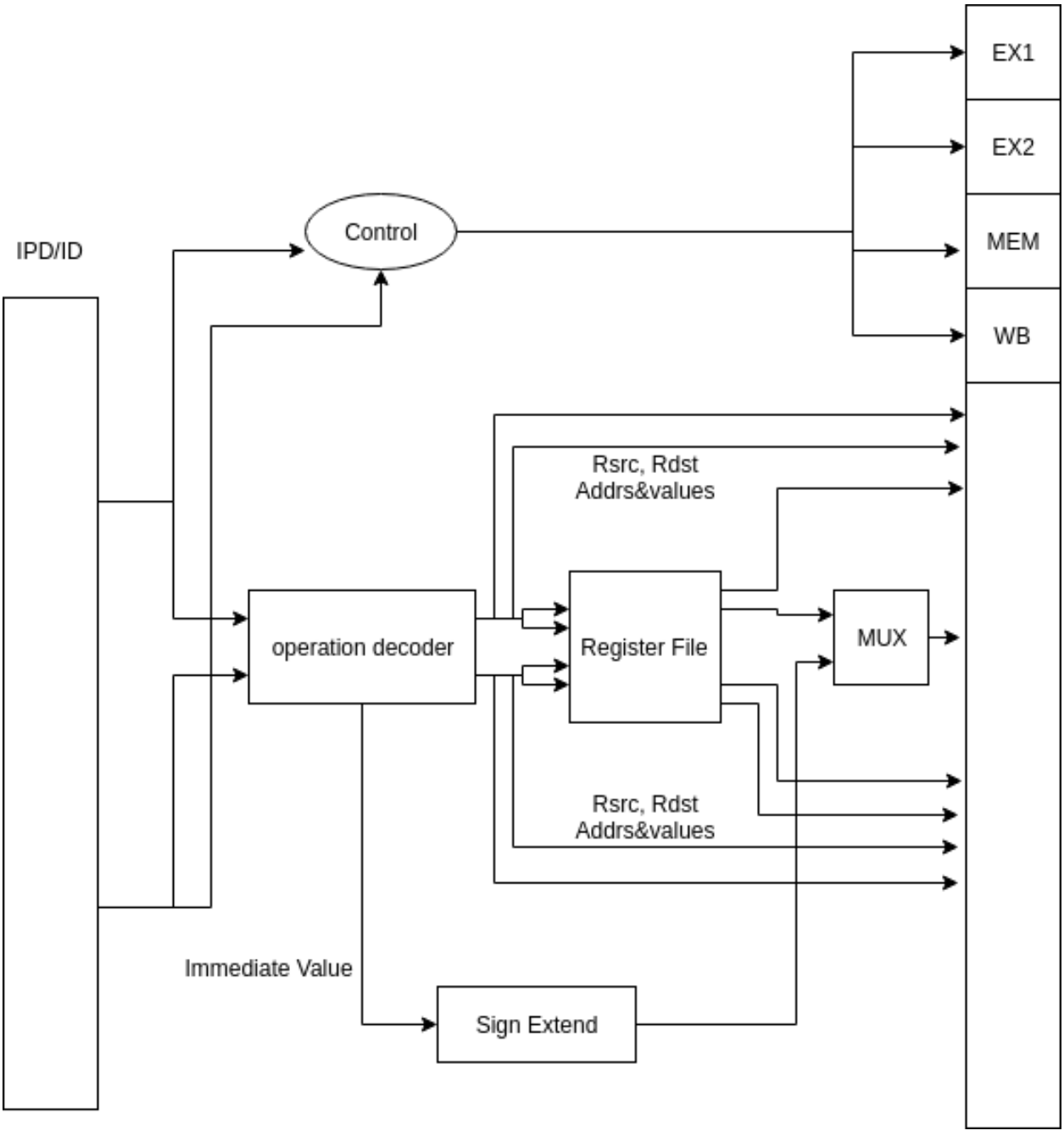

Figure 3: A schematic of the decode stage.

1. Decoding the relevant registers and reading their values. This is done through the combinational circuit 'operation decoder'. The register file allows the decoder to read four registers at once (two for each issue). It then stores their addresses and values to `ID/EX1` buffer.

2. Generates the control values for the next stages, which is done through the 'control' circuit.

3. Sign-extending the immediate value following an `LDM` instruction, storing it in the buffer's `Rsrc1` register, and giving it an appropriate `Rsrc` address.

The `ID/EX1` buffer consists of:

1. `NOP1 & NOP2` flags.

2. `EX1 & EX2 & MEM & WB` Control words.

3. `Rsrc1 & Rdst1` addresses and values.

4. `Rsrc2 & Rdst2` addresses and values.

We note that the addresses are 5-bit values. This makes it easier to address the `PC` and `SP` registers, and to indicate immediate values (not register).

2.3.1 Control Unit

The control unit is responsible for generating the control signals for `EX1, EX2, MEM` and `WB` stages. These are:

1. `EX1`: The ALU operation `ALUOp`. Update flag register `IsFlag`.

2. `EX2`: The ALU operation `ALUOp`, a bit indicating branch instruction `IsBranch`, and a bit indicating whtether to update the flag register `IsFlag`.

3. `MEM`: Index of the issue that needs the memory (if any) `MEMIssueIndex`, and operation `R/W`.

4. `WB`: Whether issue 1 or 2 (or both) require write back `IsWB1`, `IsWB2`.

2.4 Execute stage

This stage is divided into two substages. Dividing the execution into two helps solving RAW dependencies between issues inside the same packet.
The first execute stage is responsible for executing the ALU operations for the first instruction using the first ALU. The `EX1/EX2` buffer consists of:

1. `NOP1 & NOP2` flags.

2. `EX2 & MEM & WB` Control words.

3. `Rsrc1 & Rdst1` addresses and values.

4. `Rsrc2 & Rdst2` addresses and values.

The second execute stage is responsible for:

1. Executing the ALU operations for the second instruction using the second ALU. It benefits from the previous execute stage since it has the ability to have the previous ALU's output as its input through a multiplexer. Which may be considered a kind of forwarding between issues.

2. Executing branch instructions.

The `EX2/MEM` buffer consists of:

1. `NOP1 & NOP2` flags.

2. `MEM & WB` Control words.

3. `Rsrc1 & Rdst1` addresses and values.

4. `Rsrc2 & Rdst2` addresses and values.

2.5 Memory stage

It is the stage responsible for accessing the data memory, either to store or load a value. The `MEM/WB` buffer consists of:

1. `NOP1 & NOP2` flags.

2. `WB` Control word.

3. `Rsrc1 & Rdst1` addresses and values.

4. `Rsrc2 & Rdst2` addresses and values.

2.6 Write back stage

It is the final stage and the stage responsible for writing back to the register file through the interfacing circuit Register-Write block. This can write to both general registers, and special registers like `PC` and `SP`.

3 Hazards

3.1 Data Hazards

Full forwarding is used between `EX1`, `EX2`, `MEM` and `WB` stages. Although this solves many of the hazards, it doesn't solve all of them. Stalling is required for these cases:

1. Between first and fourth packet: Stall 1 cycle in the decode stage if (second issue in fourth packet is ALU instruction or any issue is memory instruction) and depends on any issue in the first packet

IF	IPD	ID	EX1	EX2	MEM	WB		
	IF	IPD	ID	EX1	EX2	MEM	WB	
		IF	IPD	ID	EX1	EX2	MEM	WB
			IF	IPD	ID	ID	EX1	EX2

2. Between first and third packet: Stall 2 cycles in the decode stage if any issue in third packet is memory instruction, and it has dependency with any issue in the first packet. Or any issue in the first packet is a load instruction, and it has dependency with the first issue in the first packet if it's an ALU instruction.

IF	IPD	ID	EX1	EX2	MEM	WB		
	IF	IPD	ID	EX1	EX2	MEM	WB	
		IF	IPD	ID	ID	ID	EX1	EX2

3. Between first and second packet: Stall 1 cycle if load-use case happens. Or if the first issue in the second packet depends on the first issue in the first packet, and both are ALU instructions.

IF	IPD	ID	EX1	EX2	MEM	WB		
	IF	IPD	IPD	ID	EX1	EX2	MEM	WB

Note that these cases can be easily translated into conditions between stages.

3.2 Branches & Control Hazards

Branches are executed in the `EX2` stage, and are always predicted to be untaken. If a branch is taken, the EX2 stage does the following:

1. Disables the load signal on the ALU flag register.
2. Sends the value of the new address to the PC generator and enables the PC generator jmp signal.
3. Flushes previous stages.

4 Opcodes

IR:

Two Operand:	5 bit opcode	3 bit reg1	3 bit reg2
One Operand:	5 bit opcode	3 bit reg	
Branch:	5 bit opcode	3 bit reg	
MEM:	5 bit opcode	3 bit reg1	3 bit reg2
MEM:	5 bit opcode	3 bit reg1	

Operation	Opcode
NOP	00 000
SETC	00 001
CLRC	00 010
NOT	00 011
INC	00 100
DEC	00 101
OUT	00 110
IN	00 111
MOV	01 000
ADD	01 001
SUB	01 010
AND	01 011
OR	01 100
SHL	01 101
SHR	01 110
PUSH	11 000
POP	11 001
LDM	11 010
LDD	11 011
STD	11 100
RESET	11 101
INT	11 110
JZ	10 000
JN	10 001
JC	10 010
JMP	10 011
CALL	10 100
RET	10 101
RETI	10 110