CMU 18-447 Introduction to Computer Architecture, Spring 2015
HW 2: ISA Tradeoffs, Microprogramming and Pipelining

Instructor: Prof. Onur Mutlu
TAs: Rachata Ausavarungnirun, Kevin Chang, Albert Cho, Jeremie Kim, Clement Loh

1 LC-3b Microcode [40 points]

Instruction	state	IRD	Cond	J	LD.MAR	LD.MDR	LD.IR	LD.BEN	LD.REG	LD.CC	LD.PC	GatePC	GateMDR	GateALU	GateMARMUX	GateSHF	PCMUX	DRMUX	SR1MUX	ADDR1MUX	ADDR2MUX	MARMUX	ALUK	MIO.EN	R.W	DATA.SIZE	LSHF1
size		1	2	6	1	1	1	1	1	1	1	1	1	1	1	1	2	1	1	1	2	1	2	1	1	1	1
BR	0	0	2	10010	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	0	X	X	X
ADD	1	0	0	10010	0	0	0	0	1	1	0	0	0	1	0	0	X	0	1	X	X	X	0	0	X	X	X
LDB	2	0	0	11101	1	0	0	0	0	0	0	0	0	0	1	0	X	X	1	1	1	1	X	0	X	X	0
STB	3	0	0	11000	1	0	0	0	0	0	0	0	0	0	1	0	X	X	1	1	1	1	X	0	X	X	0
JSR	4	0	3	10100	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	0	X	X	X
AND	5	0	0	10010	0	0	0	0	1	1	0	0	0	1	0	0	X	0	1	X	X	X	1	0	X	X	X
LDW	6	0	0	11001	1	0	0	0	0	0	0	0	0	0	1	0	X	X	1	1	1	1	X	0	X	X	1
STW	7	0	0	10111	1	0	0	0	0	0	0	0	0	0	1	0	X	X	1	1	1	1	X	0	X	X	1
RTI	8	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
XOR	9	0	0	10010	0	0	0	0	1	1	0	0	0	1	0	0	X	0	1	X	X	X	2	0	X	X	X
RES1	10	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
RES2	11	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
JMP	12	0	0	10010	0	0	0	0	0	0	1	0	0	0	0	0	2	X	1	1	0	X	X	0	X	X	X
JMP	12	0	0	10010	0	0	0	0	0	0	1	0	0	1	0	0	1	X	1	X	X	X	3	0	X	X	X
SHF	13	0	0	10010	0	0	0	0	1	1	0	0	0	0	0	1	X	0	1	X	X	X	X	0	X	X	X
LEA	14	0	0	10010	0	0	0	0	1	0	0	0	0	0	1	0	X	X	X	0	2	1	X	0	X	X	1
TRAP	15	0	0	11100	1	0	0	0	0	0	0	0	0	0	1	0	X	X	X	X	X	0	X	0	X	X	X
STW	16	0	1	10000	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	1	1	1	X
STB	17	0	1	10001	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	1	1	0	X
ALL	18	0	0	100001	1	0	0	0	0	0	1	1	0	0	0	0	0	X	X	X	X	X	X	0	X	X	X
ALL	18	0	0	100001	1	0	0	0	0	0	1	0	0	0	1	0	X	X	X	0	0	1	X	0	X	X	X
ALL	19	0	0	100001	1	0	0	0	0	0	1	1	0	0	0	0	0	X	X	X	X	X	X	0	X	X	X
ALL	19	0	0	100001	1	0	0	0	0	0	1	0	0	0	1	0	X	X	X	0	0	1	X	0	X	X	X
JSR	20	0	0	10010	0	0	0	0	1	0	1	1	0	0	0	0	2	1	1	1	0	X	X	0	X	X	X
JSR	20	0	0	10010	0	0	0	0	1	0	1	1	0	0	0	0	2	1	1	1	1	X	X	0	X	X	X
JSR	21	0	0	10010	0	0	0	0	1	0	1	1	0	0	0	0	2	1	X	0	3	X	X	0	X	X	1
BR	22	0	0	10010	0	0	0	0	0	0	1	0	0	0	0	0	2	X	X	0	2	X	X	0	X	X	1
STW	23	0	0	10000	0	1	0	0	0	0	0	0	0	1	0	0	X	X	0	X	X	X	3	0	X	X	X
STW	23	0	0	10000	0	1	0	0	0	0	0	0	0	0	1	0	X	X	0	1	0	1	X	0	X	X	X
STB	24	0	0	10001	0	1	0	0	0	0	0	0	0	1	0	0	X	X	0	X	X	X	3	0	X	X	X
STB	24	0	0	10001	0	1	0	0	0	0	0	0	0	0	1	0	X	X	0	1	0	1	X	0	X	X	X
LDW	25	0	1	11001	0	1	0	0	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	1	0	X	X
FREE	26	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
LDW	27	0	0	10010	0	0	0	0	1	1	0	0	1	0	0	0	X	0	X	X	X	X	X	0	X	1	X
TRAP	28	0	1	11100	0	1	0	0	1	0	0	1	0	0	0	0	X	1	X	X	X	X	X	1	0	X	X
LDB	29	0	1	11101	0	1	0	0	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	1	0	X	X
TRAP	30	0	0	10010	0	0	0	0	0	0	1	0	1	0	0	0	1	X	X	X	X	X	X	0	X	1	X
LDB	31	0	0	10010	0	0	0	0	1	1	0	0	1	0	0	0	X	0	X	X	X	X	X	0	X	0	X
ALL	32	1	X	X	0	0	0	1	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	0	X	X	X
ALL	33	0	1	100001	0	1	0	0	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	1	0	X	X
FREE	34	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
ALL	35	0	0	100000	0	0	1	0	0	0	0	0	1	0	0	0	X	X	X	X	X	X	X	0	X	1	X

2 Addressing Modes [12 points]

(a) Auto increment

(b) Scale indexed

(c) Register indirect

(d) Memory indirect

3 Microarchitecture vs. ISA [15 points]

(a) The ISA level is the interface a machine exposes to the software. The microarchitecture is the actual underlying implementation of the machine. Therefore, the microarchitecture and changes to the microarchitecture are transparent to the compiler/programmer (except in terms of performance), while changes to the ISA affect the compiler/programmer.

The compiler does *not* need to know about the microarchitecture of the machine in order to compile the program correctly.

(b) (i) ISA

(ii) Microarchitecture

(iii) ISA

(iv) ISA

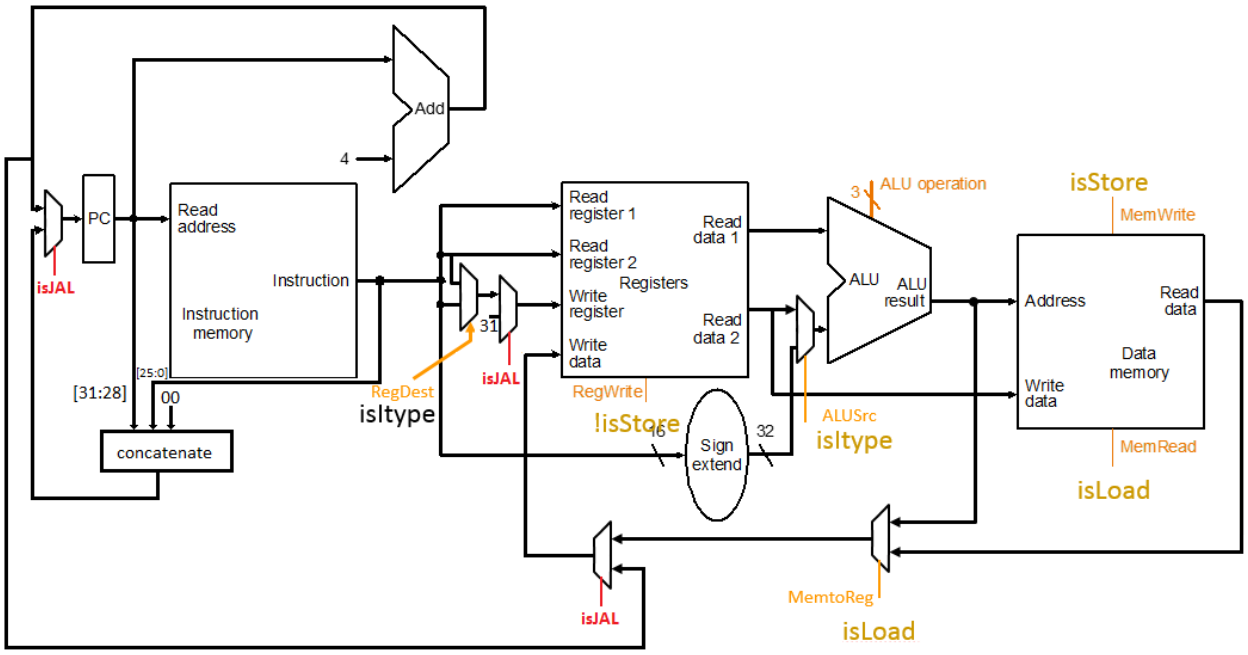
(v) Microarchitecture

(vi) ISA

(vii) Microarchitecture

(viii) Microarchitecture

4 Single-Cycle Processor Datapath [30 points]



5 Pipelining [30 points]

(a) A non-pipelined machine

```
9 + 7 + 7 + 9 + 7 + 9 = 48 cycles
```

(b) A pipelined machine with scoreboarding and five adders and five multipliers without data forwarding

```
Cycles          1|2|3|4|5|6|7|8|9|10|11|12|13|14|15|16|17|18|19|20|21|22|23|24|25|26
MUL R3, R1, R2  F|D|E|E|E|E|E|E|W
ADD R5, R4, R3    F|D|-|-|-|-|-|-|D |E |E |E |E |W
ADD R6, R4, R1      F|-|-|-|-|-|-|- |D |E |E |E |E |W
MUL R7, R8, R9                      F |D |E |E |E |E |E |E |W
ADD R4, R3, R7                         F |D |- |- |- |- |- |- |D |E |E |E |E |W
MUL R10, R5, R6                           F |- |- |- |- |- |- |- |D |E |E |E |E |E |E |W
```

28 cycles (or 26 cycles with internal register file data forwarding)

(c) A pipelined machine with scoreboarding and five adders and five multipliers with data forwarding.

```
Cycles          1|2|3|4|5|6|7|8|9|10|11|12|13|14|15|16|17|18|19|20|21|22|23|24
MUL R3, R1, R2  F|D|E|E|E|E|E|E|W
ADD R5, R4, R3    F|D|-|-|-|-|-|E|E |E |E |W
ADD R6, R4, R1      F|-|-|-|-|-|D|E |E |E |E |W
MUL R7, R8, R9                  F|D |E |E |E |E |E |E |W
ADD R4, R3, R7                    F |D |- |- |- |- |- |E |E |E |E |W
MUL R10, R5, R6                      F |- |- |- |- |- |D |E |E |E |E |E |E |W
```

24 cycles

(d) A pipelined machine with scoreboarding and one adder and one multiplier without data forwarding

```
Cycles          1|2|3|4|5|6|7|8|9|10|11|12|13|14|15|16|17|18|19|20|21|22|23|24|25|26|27|28|29|30|31
MUL R3, R1, R2  F|D|E|E|E|E|E|E|W|
ADD R5, R4, R3    F|D|-|-|-|-|-|-|D |E |E |E |E |W
ADD R6, R4, R1      F|-|-|-|-|-|-|- |D |- |- |D |E |E |E |E |W
MUL R7, R8, R9                      F |- |- |- |D |E |E |E |E |E |E |W
ADD R4, R3, R7                                  F |D |- |- |- |- |- |- |D |E |E |E |E |W
MUL R10, R5, R6                                    F |- |- |- |- |- |- |- |D |E |E |E |E |E |E |W
```

31 cycles (or 29 cycles with internal register file data forwarding)

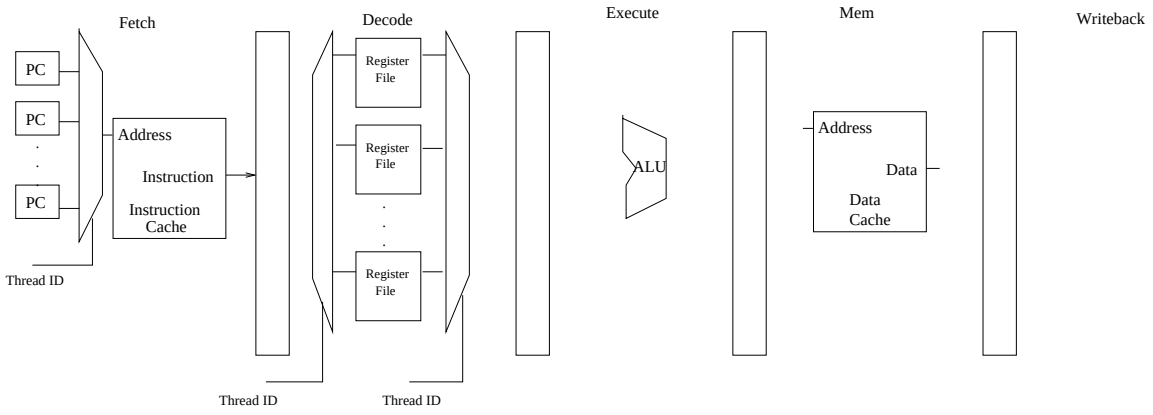
(e) A pipelined machine with scoreboarding and one adder and one multiplier with data forwarding

```
Cycles          1|2|3|4|5|6|7|8|9|10|11|12|13|14|15|16|17|18|19|20|21|22|23|24|25|26|27
MUL R3, R1, R2  F|D|E|E|E|E|E|E|W|
ADD R5, R4, R3    F|D|-|-|-|-|-|E|E |E |E |W
ADD R6, R4, R1      F|-|-|-|-|-|D|- |- |- |E |E |E |E |W
MUL R7, R8, R9                  F|- |- |- |D |E |E |E |E |E |E |W
ADD R4, R3, R7                            F |D |- |- |- |- |- |E |E |E |E |W
MUL R10, R5, R6                              F |- |- |- |- |- |D |E |E |E |E |E |E |W
```

27 cycles

6 Fine Grain Multi-threading [40 points]

(a) The figure shows the solution



(b)

3

Why?

Since branches are resolved in the Execute stage, it is necessary that the Fetch stage does not fetch for a thread until the thread's previous instruction has passed Execute. Hence three threads are needed to cover Fetch, Decode, Execute.

(c)

4

Why?

The designer must ensure that when an instruction is in Writeback, the next instruction in the same thread has not reached Decode yet. Hence, at least 4 threads are needed.

(d) Is the number of threads required to eliminate branch-related stalls in Machine II the same as in Machine I?

YES **NO** (Circle one)

If yes, why?

Branches are resolved at the third pipeline stage in both machines, and distance from fetch to branch resolution determines the minimum number of threads to avoid branch stalls.

(e)

3 (if no flow dependence stalls occur)

(f) Does Machine II require the same minimum number of threads as Machine I to avoid the need for flow-dependence stalls?

YES **NO** (Circle one)

how many threads are required?

12 (the Decode, Execute 1 – 8, Memory, and Writeback stages must all have instructions from independent threads.)

(g)

12

(h)

The additional FGMT-related logic (MUXes and thread selection logic) could increase the critical path length, which will reduce maximum frequency and thus performance.

7 Branch Prediction and Dual Path Execution [35 points]

(a) 5 instructions.

(b) *Note that if you assumed the wrong number of instructions in Part (a), you will only be marked wrong for this in Part (a). You can still get full credit on this and the following parts.*

$$\begin{aligned}
\text{Correct path instructions} &= N \\
\text{Incorrect path instructions} &= N(0.2)(1-A)5 = N(1-A) \\
\text{Fetched instructions} &= \text{Correct path instructions} + \text{Incorrect path instructions} \\
&= N + N(1-A) \\
&= \boxed{N(2-A)}
\end{aligned}$$

(c)

$$\begin{aligned}
\text{Correct path instructions} &= N \\
\text{Incorrect path instructions} &= N(0.2)5 \\
\text{Fetched instructions} &= \text{Correct path instructions} + \text{Incorrect path instructions} \\
&= N + N(1-0.8)5 \\
&= \boxed{2N}
\end{aligned}$$

This solution assumes you have enough hardware in the frontend of the machine to fetch concurrently from both paths. If you assumed that both paths are fetched from on alternate cycles, that high-level approach is also OK, although note that you would need additional branch taken and not taken information to solve it completely.

(d)

$$
\begin{aligned}
\text{Correct path instructions} &= N \\
\text{Incorrect path instructions due to...} & \\
&\quad \text{lack of confidence} \\
&\quad\quad = N(0.2)(1-C)5 = N(1-C) \\
&\quad \text{incorrect high confidence estimate} \\
&\quad\quad = N(0.2)CM5 = NCM \\
\text{Fetched instructions} &= \text{Correct path instructions} \\
&\quad + \text{Incorrect path instructions due to} \\
&\quad\quad \text{lack of confidence} \\
&\quad + \text{Incorrect path instructions due to} \\
&\quad\quad \text{incorrect high confidence estimate} \\
&= N + N(1-C) + NCM \\
&= \boxed{N[2 + C(M-1)]}
\end{aligned}
$$

Like above, if you assumed a different execution model for Part (c), you will not be penalized for using it in this part.

8 Mysterious Instruction [40 points]

This instruction does a vector increment.