CMU 18-447 Introduction to Computer Architecture, Spring 2013
HW 2 Solutions: ISA Tradeoffs

Instructor: Prof. Onur Mutlu
TAs: Justin Meza, Yoongu Kim, Jason Lin

1 LC-3b Microcode [80 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 Data Reuse in Different ISAs [30 points]

(a) **Assembly code sequences for code segments A and B for machines X and Y**

Code Segment A
Machine X

```
ADD A, A, 1
ADD B, B, 2
ADD C, C, 3
ADD D, D, 4
```

Machine Y

```
LD R1, A
ADD R1, R1, 1
ST R1, A
LD R2, B
ADD R2, R2, 2
ST R2, B
LD R3, C
ADD R3, R3, 3
ST R3, C
LD R4, D
ADD R4, R4, 4
ST R4, D
```

Code Segment B
Machine X

```
ADD B, B, A
ADD C, B, A
SUB A, B, C
SUB D, A, C
```

Machine Y

```
LD R1, A
LD R2, B
ADD R3, R1, R2
ADD R4, R3, R1
SUB R5, R3, R4
SUB R6, R5, R4
ST R3, B
ST R4, C
ST R5, A
ST R6, D
```

(b) **Total number of bytes transferred**

	Code Segment A	Code Segment B
Machine X	60	76
Machine Y	80	64

(c) (i) No. For code segment A, machine X has lower number of total bytes transferred. For code segment B, machine Y has lower number of total bytes transferred.

(ii) Code segment A has no data reuse. It just adds an immediate value to A, B and C each, and does not reuse the values of A, B or C. A memory-memory machine needs just three instructions to express code segment A. Each instruction fetches a memory location, adds an immediate value to it and stores it back. On the other hand, a load-store machine incurs several instructions to load the values of A, B and C into registers and store them back. This is unnecessary and merely increases the number of instructions (and hence number of instruction bytes transferred) for code segment A, as there is no reuse of data.

Code segment B reuses the values of A and B. In this case, a load-store machine loads A and B into registers first and reuses the values stored in registers. On the other hand, a memory-memory machine does repeated memory accesses to memory to fetch A and B every time they are used. Therefore, a load-store machine reduces the number of data bytes transferred significantly, for code segment B, resulting in a smaller number of total bytes transferred.

3 Fixed- vs. Variable-Length ISAs [20 points]

(a) Fixed Length ISA - $4 \times 4 = 16$ bytes
Variable Length ISA - 1×4 (ADD instruction) $+3 \times 3$ (other instructions) $= 13$ bytes

(b) Fixed Length ISA - 3×1 (other instructions) $+1 \times 4$ (STW instruction) $= 7$ cycles
Variable Length ISA - 3×3 (other instructions) $+1 \times 6$ (STW instruction) $= 15$ cycles

(c) The variable length ISA has lower code size, because it uses fewer bytes to encode the instructions in the given code segment.

(d) The fixed length ISA has lower execution time, because the decode complexity and hence the number of cycles to decode an instruction is smaller.

4 Addressing Modes [12 points]

(a) Auto increment

(b) Scale indexed

(c) Register indirect

(d) Memory indirect

5 Addressability [15 points]

(a) (i) 29 bits
(ii) 26 bits
(iii) 23 bits
(iv) 21 bits

(b)
```
# load the two words
lw   $8, 0($5)
lw   $9, 0($7)

# multiply the offset by 4 to get the shift amount
sll  $4, $4, 2
sll  $6, $6, 2

# shift the loaded word to obtain the nibble at the desired offset
srlv $8, $8, $4
srlv $9, $9, $6
```

```
# zero out the rest of the word to keep the nibble
andi $8, $8, 0xF
andi $9, $9, 0xF

add  $2, $8, $9
```

6 Microarchitecture vs. ISA [20 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

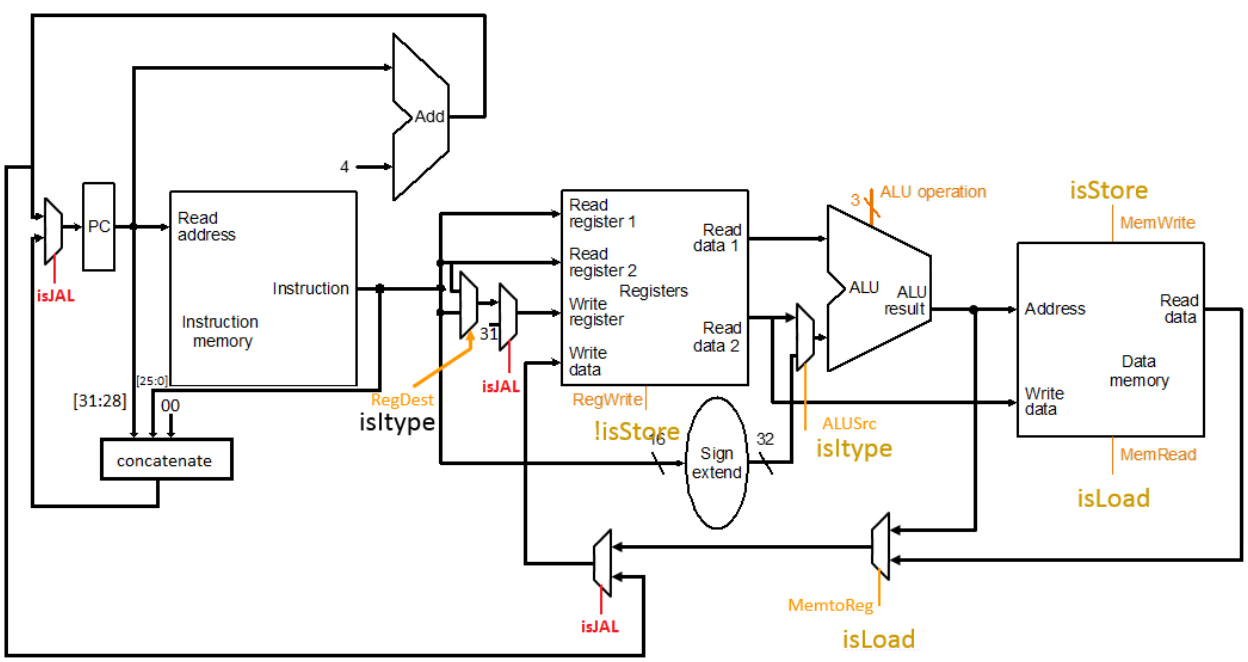
(iii) ISA

(iv) ISA

(v) Microarchitecture

(vi) ISA

(vii) Microarchitecture

(viii) Microarchitecture

7 Single-Cycle Processor Datapath [30 points]