0907432 Computer Design (Spring 2020)
Homework

رقم التسجيل: الاسم:

==

Instructions: The deadline is **Sunday 3/5/2020 at 8:00 am**. There are six problems and each problem is for 2 marks. Please answer all problems with **clear handwriting**; typed answers are **not** acceptable. Each student must submit her/his own answers; submitting multiple similar answers is considered cheating and disciplinary actions will be taken. **Scan your answer sheets** using an appropriate smartphone app like Microsoft Office Lens. It is best that you assemble all your answer images in **one pdf file** and **email it to abandah@ju.edu.jo**. You must send your answers **from your university email** that ends @ju.edu.jo.

<Good Luck>

P1. A multicore processor has a clock rate of 3.0 GHz and voltage of 1.0 V. Assume that, on average, it consumes 75 W of static power and 30 W of dynamic power when all its six cores are active. It executes a program consisting of 10^{10} instructions on a single core with the following instruction mix.

Instruction Type	CPI	Frequency
Arithmetic	1.0	50%
Load/store	2.0	40%
Branch instructions	1.5	10%

a) Find the total execution time for this program on one core.

CPU Time $= IC \times \Sigma\ (CPI \times freq) \div clock\ rate$
$= 10^{10} \times (1.0 \times 0.5 + 2.0 \times 0.4 + 1.5 \times 0.1) \div 3 \times 10^{9}$
$= 10 \times (0.5 + 0.8 + 0.15) \div 3$
$= 10 \times 1.45 \div 3 = 4.833$ **seconds**

b) When this program is parallelized and run on the six cores, the number of arithmetic and load/store instructions per core is divided by 4, but the number of branch instructions remains the same. What is the execution time of the parallel program?

Exec. Time $= IC \times \Sigma\ (CPI \times freq) \div clock\ rate$
$= 10^{10} \times (1.0 \times 0.5 \div 4 + 2.0 \times 0.4 \div 4 + 1.5 \times 0.1) \div 3 \times 10^{9}$
$= 10 \times (0.125 + 0.2 + 0.15) \div 3$
$= 10 \times 0.475 \div 3 = 1.583$ **seconds**

c) What is the speedup of the parallel program?

Speedup $= 4.833 \div 1.583 = 3.05$

d) Assumes that the power consumed is proportional to the number of active cores. What are the energies consumed by the serial program on the single core and the parallel program on the six cores?

Energy of the serial program = Exec. Time × Power
$= 4.833 \times (75 + 30) \div 6$
= 85 **Joules**

Energy of the parallel program = Exec. Time × Power
$= 1.583 \times (75 + 30)$
= 166 **Joules**

e) What is the average capacitive load of this processor when the six cores are active?

Dynamic Power $= \frac{1}{2} \times$ Capacitive load $\times$ Voltage$^2 \times$ Frequency
30 $= \frac{1}{2} \times$ Capacitive load $\times 1.0^2 \times 3\times10^9$
Capacitive load $= 2 \times 30 \div 3\times10^9 = 20$ **nF (nano Farads)**

f) What is the average current drawn by this processor when the six cores are active?

Power = Current × Voltage
Current $= (75+30) \div 1.0 = 105$ **Amperes**

P2. The following loop is executed for 1,000 iterations.

```
Loop: ld   x10, 0(x13)
      ld   x11, 0(x14)
      mul  x12, x10, x11
      add  x16, x16, x12
      sd   x12, 0(x17)
      addi x17, x17, 8
      addi x14, x14, 8
      addi x13, x13, 8
      beq  x13, x12, Loop
```

a) Assume that this loop is executed on the five-stage RISC-V pipelined processor studied in the class. Assume also that this processor solves all data hazards through forwarding and stalls, resolves the branch instructions in the decode stage, and has one-cycle branch delay. How many cycles are needed to execute the 1,000 iterations ignoring the time needed to fill the pipeline?

```
                        1 2 3 4 5 6 7 8 9 0 1 2
ld   x10, 0(x13)        F D E M W
ld   x11, 0(x14)          F D E M W
mul  x12, x10, x11          F D D E M W
add  x16, x16, x12            F F D E M W
sd   x12, 0(x17)                  F D E M W
addi x17, x17, 8                    F D E M W
addi x14, x14, 8                      F D E M W
addi x13, x13, 8                        F D E M W
beq  x13, x12, Loop                       F D D E M W
ld   x10, 0(x13)                                F ...
```

From the pipeline diagram above, we see that it takes 12 cycles to finish one iteration and to start fetching instructions of the next iteration.
Also, time of one iterations = 9 instrs. + 2 stall cycles + 1 branch delay = 12 cycles
Total time = 12 × 1,000 = 12,000 cycles

b) Unroll this loop two times and schedule it for the static dual-issue processor studied in the class?
Performing replication, removing loop overhead, ad modifying instructions, and register renaming:

```
Loop: ld   x10, 0(x13)
      ld   x11, 0(x14)
      mul  x12, x10, x11
      add  x16, x16, x12
      sd   x12, 0(x17)
      ld   x20, 8(x13)
      ld   x21, 8(x14)
      mul  x22, x20, x21
      add  x16, x16, x22
      sd   x22, 8(x17)
      addi x17, x17, 16
      addi x14, x14, 16
      addi x13, x13, 16
      beq  x13, x12, Loop
```

Scheduling:

	ALU/branch	Load/store	Cycle
`Loop:`	`addi x17, x17, 16`	`ld x10, 0(x13)`	1
	`addi x14, x14, 16`	`ld x11, 0(x14)`	2
	`mul x12, x10, x11`	`ld x20, 8(x13)`	3
	`add x16, x16, x12`	`ld x21, -8(x14)`	4
	`addi x13, x13, 16`	`sd x12, -16(x17)`	5
	`mul x22, x20, x21`		6
	`add x16, x16, x22`	`sd x22, -8(x17)`	7
	`beq x13, x12, Loop`		8

c) Assuming same conditions as in (a) above, how many cycles are needed to execute the unrolled loop on the dual-issue processor?

It takes 8 cycles to issue two iterations.
Total time = (8+1) × (1,000 ÷ 2) = 4,500 cycles

P3. Assume that the following loop is executed by a speculative pipelined processor of degree 3. This processor uses reservation stations, common data buses, and reorder buffer. The fetch stage takes one cycle and the issue stage takes one cycle. The integer/branch latency is 1 cycle and the load latency is 2 cycles (1 cycle for address calculation and 1 cycle for data memory access). The processor has two address calculation units, two memory access units, two integer ALU units, and one branch unit.

```
Loop: ld   x10, 0(x13)
      ld   x11, 0(x14)
      mul  x12, x10, x11
      add  x16, x16, x12
      sd   x12, 0(x17)
      addi x17, x17, 8
      addi x14, x14, 8
      addi x13, x13, 8
      beq  x13, x12, Loop
```

a) Assume that branch prediction is used and the branch instruction is predicted wrongly as not taken at the end of the first iteration. Draw the multi-cycle pipeline diagram for the execution of the first two iterations of this loop to find how many cycles are needed to fetch and commit the two iterations.

```
                    1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
ld   x10, 0(x13)    F I A M W C
ld   x11, 0(x14)    F I A M W C
mul  x12, x10, x11  F I       E W C
add  x16, x16, x12    F I         E W C
sd   x12, 0(x17)      F I A           C
addi x17, x17, 8      F I E W         C
addi x14, x14, 8        F I E W         C
addi x13, x13, 8        F I E W         C
beq  x13, x12, Loop     F I E W         C
I10                       F . . .         n
. . .
ld   x10, 0(x13)                              F I A M W C
ld   x11, 0(x14)                              F I A M W C
mul  x12, x10, x11                            F I       E W C
add  x16, x16, x12                              F I         E W C
sd   x12, 0(x17)                                F I A           C
addi x17, x17, 8                                F I E W         C
addi x14, x14, 8                                  F I E W         C
addi x13, x13, 8                                  F I E W         C
beq  x13, x12, Loop                               F I E W         C
```

22 cycles for two iterations.

b) Now assume that this processor has perfect branch prediction with no branch delay. Estimate the number of cycles needed to execute 1,000 iterations of this loop ignoring the time needed to fill the pipeline? (Show your work clearly)

```
                    1 2 3
ld   x10, 0(x13)    F I A M W C
ld   x11, 0(x14)    F I A M W C
mul  x12, x10, x11  F I       E W C
add  x16, x16, x12    F I         E W C
sd   x12, 0(x17)      F I A           C
addi x17, x17, 8      F I E W         C
addi x14, x14, 8        F I E W         C
addi x13, x13, 8        F I E W         C
beq  x13, x12, Loop     F I E W         C
ld   x10, 0(x13)          F I A M W       C
ld   x11, 0(x14)          F I A M W       C
mul  x12, x10, x11        F I       E W   C
add  x16, x16, x12          F I         E W C
sd   x12, 0(x17)            F I A           C
addi x17, x17, 8            F I E W         C
addi x14, x14, 8              F I E W         C
addi x13, x13, 8              F I E W         C
beq  x13, x12, Loop           F I E W         C
```

3 cycles are needed to fetch and issue each iteration.
Total time = 3 × 1,000 = 3,000 cycles

P4. Assume you have a two-way set associative cache with four-word blocks and a total size of 32 words. Assume also that the cache is initially empty and uses true LRU replacement policy. For the following sequence of word-address references (not byte addresses), trace by completing the table below the behavior of the cache. For each reference, identify: the binary word address, the tag, the index, the word offset, whether the reference is a hit or a miss (compulsory, conflict or capacity), and which tags are in each way of the cache after the reference has been handled.

`0x03, 0xb4, 0x2b, 0x02, 0xbe, 0x58, 0xbf, 0x0e, 0x1f, 0xbc`

4 words per block → 2 bits for the word offset
Number of blocks = 32 words / 4 = 8 blocks
Number of sets = 8 blocks / 2 ways = 4 sets → 2 bits for the index

Word Address	Binary Address	Tag	Index	Word Offset	Hit/Miss	Miss Type	Way 0	Way 1
0x03	00000011	0	0	3	M	Comp.	**T(0)=0** T(1)=- T(2)=- T(3)=-	T(0)=- T(1)=- T(2)=- T(3)=-
0xb4	10110100	b	1	0	M	Comp.	**T(0)=0** **T(1)=b** T(2)=- T(3)=-	T(0)=- T(1)=- T(2)=- T(3)=-
0x2b	00101011	2	2	3	M	Comp.	**T(0)=0** **T(1)=b** **T(2)=2** T(3)=-	T(0)=- T(1)=- T(2)=- T(3)=-
0x02	00000010	0	0	2	H		**T(0)=0** **T(1)=b** **T(2)=2** T(3)=-	T(0)=- T(1)=- T(2)=- T(3)=-
0xbe	10111110	b	3	2	M	Comp.	**T(0)=0** **T(1)=b** **T(2)=2** **T(3)=b**	T(0)=- T(1)=- T(2)=- T(3)=-
0x58	01011000	5	2	0	M	Comp.	**T(0)=0** **T(1)=b** T(2)=2 **T(3)=b**	T(0)=- T(1)=- **T(2)=5** T(3)=-
0xbf	10111111	b	3	3	H		**T(0)=0** **T(1)=b** T(2)=2 **T(3)=b**	T(0)=- T(1)=- **T(2)=5** T(3)=-
0x0e	00001110	0	3	2	M	Comp.	**T(0)=0** **T(1)=b** T(2)=2 T(3)=b	T(0)=- T(1)=- **T(2)=5** **T(3)=0**
0x1f	00011111	1	3	3	M	Comp.	**T(0)=0** **T(1)=b** T(2)=2 **T(3)=1**	T(0)=- T(1)=- **T(2)=5** T(3)=0
0xbc	10111100	b	3	0	M	Conf.	**T(0)=0** **T(1)=b** T(2)=2 T(3)=1	T(0)=- T(1)=- **T(2)=5** **T(3)=b**

*** Bold indicates most recently accessed way.**

P5. Draw a diagram that shows the TLB and cache interaction of a system with the following specifications: 32-bit virtual and physical addresses, 4-KB pages, 8-entry fully-associative TLB, and 4-way associative, physically-tagged, 16-Kbyte cache of 64-byte blocks. You must show the widths of all busses and the TLB and the cache hit circuits.

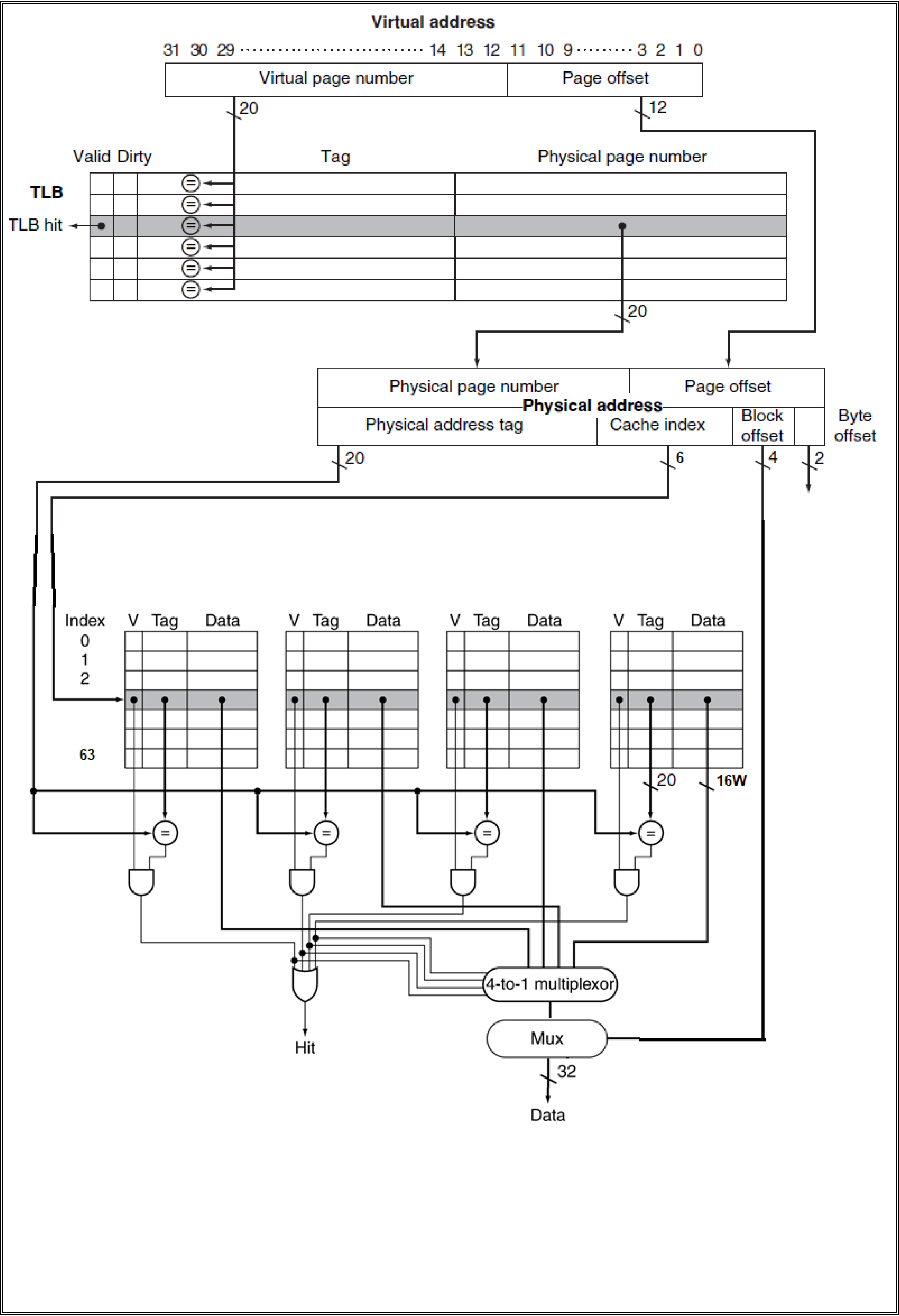
<page offset> = $\lg_2$ 4K = 12 bits

<block offset> = $\lg_2$ 64 = 6 bits

Number of cache sets = 16 Kbytes / 64 bytes / 4 ways = 64 sets

<index> = $\lg_2$ 64 = 6 bits

<tag> = 32 – 6 – 6 = 20 bits


Virtual address
31 30 29 14 13 12 11 10 9 3 2 1 0
Virtual page number
Page offset
20
12
Valid Dirty
Tag
Physical page number
TLB
TLB hit
20
Physical page number
Page offset
Physical address
Physical address tag
Cache index
Block offset
Byte offset
20
6
4
2
Index
V
Tag
Data
0
1
2
63
20
16W
4-to-1 multiplexor
Mux
32
Hit
Data

P6. Dependability

a) The following figure shows the parity bits, data bits, and field coverage in a Hamming ECC code for eight data bits. Assume that you have a memory module that uses this ECC code and that the binary sequence `110001000101` is read from the memory and it has a single bit error. What are the original eight data bits?

Bit position		1	2	3	4	5	6	7	8	9	10	11	12
Encoded data bits		p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8
Parity bit coverage	p1	X		X		X		X		X		X	
	p2		X	X			X	X			X	X	
	p4				X	X	X	X					X
	p8								X	X	X	X	X

```
123456789012
110001000101
ppdpdddpdddd

The data bits are 0010 0101

p1 = xor(1 0 0 0 0 0) = 1
p2 = xor(1 0 1 0 1 0) = 1
p4 = xor(0 0 1 0 1) = 0
p8 = xor(0 0 1 0 1) = 0

The error code is 0011 -> error in bit 3 (d1).

So the data is 1010 0101
```

b) Who uses RAID 51 and why?

Organizations that have branches in multiple locations use RAID 51.
Compared with RAID 5, RAID 51 offers higher throughput, lower local latency, and higher redundancy and availability.