# LSP Exam Cram Guide (EN)

Author: LSP Study Notes Maintainers

## 1. Before the Exam

**[HACK] Exam Survival Rules**

**Remember three rules:**

1. **Skip what you can't do first** and secure the easy points
2. **Write the steps for partial credit**; even with a wrong final number, you can still get process points
3. **If you're running out of time, guess**; leaving it blank is guaranteed 0

### 1.1 Exam Scope (as the instructor said)

**Must-test vs Not tested**

**Must-test:**

- Signed/unsigned arithmetic
- 4-variable K-Map simplification
- RS latch waveforms
- Shannon expansion
- Pipeline hazards

**Not tested:** ~~Cache calculations~~, ~~branch predictor~~

**Source:** Instructor email: "Branch prediction and cache are not tested; pipeline will be tested."

### 1.2 Recommended Solving Order

**Suggested order**

1. **Number problems** (15min) - easiest, lock points early
2. **K-Map** (20min) - drawing task, be careful
3. **Pipeline** (20min) - follow the template
4. **RS/Shannon** (25min) - do it step by step
5. **Check** (10min) - verify calculations

## 2. Number Systems (Calculator-first)

**[HACK] Core idea: compute in decimal, convert at the end**

**Don't waste time doing the whole thing in binary.**

Compute in decimal first, and handle overflow only at the last step.

### 2.1 Powers of Two (memorize)

| | | | | | |
|---|---|---|---|---|---|
| $2^4$ | 16 | $2^8$ | 256 | $2^{12}$ | 4096 |
| $2^5$ | 32 | $2^9$ | 512 | $2^{14}$ | 16384 |
| $2^6$ | 64 | $2^{10}$ | 1024 | $2^{16}$ | 65536 |
| $2^7$ | 128 | $2^{11}$ | 2048 | $2^{20}$ | 1M |

### 2.2 N-bit ranges (must know)

**Range quick notes**

- **Unsigned:** $[0, 2^N - 1]$
- **Signed:** $[-2^{N-1}, 2^{N-1} - 1]$

**Common:**

- 8-bit signed: $[-128, +127]$
- 10-bit signed: $[-512, +511]$
- 16-bit signed: $[-32768, +32767]$

### 2.3 Unsigned overflow

**[HACK] Unsigned overflow - take modulo**

**Rule:** result $=$ (decimal result) $\%\ 2^N$

**Example:** 8-bit unsigned, compute $200 + 100$

1. Decimal: $200 + 100 = 300$
2. Modulo: $300 \mod 256 = 300 - 256 = 44$
3. **Answer: 44**

That's it. No need to convert to binary.

### 2.4 Signed overflow

**[HACK] Signed overflow - subtract $2^N$ if above max**

**Three-step recipe:**

1. Compute the decimal result $R$
2. Check: $R > 2^{N-1} - 1$ (max positive)?
3. If yes: $R - 2^N$ is the final answer

**Example:** 10-bit signed, compute $511 + 511$

1. Decimal: $511 + 511 = 1022$
2. Check: $1022 > 511$? Yes, overflow.
3. Fix: $1022 - 1024 = -2$
4. **Answer:** $-2$

**Symmetric case:** if $R < -2^{N-1}$, then add $2^N$.

### 2.5 Negative numbers to two's complement

**[HACK] Two's complement for negatives - no bit-flip + 1 needed**

**Pro formula:** two's complement of $-X$ is $2^N - X$

**Example:** 8-bit representation of $-5$

1. Compute: $256 - 5 = 251$

2. Convert to binary: 251 = 128+64+32+16+8+2+1
3. **Answer:** `11111011`

Much faster than "invert bits + 1".

### 2.6 Two's complement to decimal

**[HACK] Use MSB to determine sign**

**Check the most significant bit:**

- MSB=0: positive, convert normally
- MSB=1: negative, use the formula

**Negative formula:** value = (unsigned binary value) $-2^N$

**Example:** 8-bit `11110100`

1. MSB=1, so it's negative
2. Treat as unsigned: $128 + 64 + 32 + 16 + 4 = 244$
3. Subtract $2^8$: $244 - 256 = -12$
4. **Answer:** $-12$

### 2.7 Sign extension

**[HACK] Extending bit-width - copy the sign bit**

**Unsigned:** pad with 0s

**Signed:** replicate MSB

- Positive (MSB=0): pad with 0s
- Negative (MSB=1): pad with 1s

**Example:** 4-bit → 8-bit

- `0110` (+6) → `0000 0110`
- `1010` (-6) → `1111 1010`

**[!] Don't lose free points**

- Read carefully: signed vs unsigned
- Watch the bit-width (8-bit vs 10-bit ranges differ)
- Use the right powers of two: $2^{10} = 1024$, not 1000

## 3. Karnaugh Map (spot-the-difference)

**[HACK] K-Map is just a spot-the-difference game**

**Three steps:**

1. **Fill the map:** set cells to 1 according to minterms
2. **Group the 1s:** circle them (bigger is better)
3. **Write terms:** for each group, write the variables that do not change

### 3.1 4-variable K-Map template (copy it)

| $AB$ \ $CD$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 1 | 3 | 2 |
| 01 | 4 | 5 | 7 | 6 |
| 11 | 12 | 13 | 15 | 14 |
| 10 | 8 | 9 | 11 | 10 |

**Gray code order: 00, 01, 11, 10** (memorize)

### 3.2 How to fill the map

**[HACK] Where does each variable go?**

**Regions for ABCD in a 4-variable map:**

- $A = 1$: bottom two rows (row 11, 10)
- $B = 1$: middle two rows (row 01, 11)
- $C = 1$: middle two columns (col 01, 11)
- $D = 1$: right two columns (col 11, 10)

**Negation:** $\bar{A}$ means the $A = 0$ region.

### 3.3 Grouping with no-brain patterns

**[HACK] Four must-know grouping patterns**

1. **Four corners** = $\bar{B}\bar{D}$

| | | | |
|---|---|---|---|
| 1 | | | 1 |
| | | | |
| | | | |
| 1 | | | 1 |

2. **Whole row** = only AB changes

| | | | |
|---|---|---|---|
| | | | |
| 1 | 1 | 1 | 1 |
| | | | |
| | | | |

3. **2x2 block (4 cells)** = eliminates two variables
4. **Wrap-around (top/bottom or left/right)** = groups may cross edges

### 3.4 Grouping rules

**[!] Three iron rules**

1. Group size must be a **power of two**: 1, 2, 4, 8, 16
2. **Bigger is better** (eliminate more variables)
3. Every 1 must be **covered at least once**

### 3.5 Write the expression from a group

**[HACK] Write what does not change inside the group**

**Look at variable values within the group:**

- always 1 → write the variable
- always 0 → write the negated variable
- sometimes 0 sometimes 1 → omit it (it cancels)

**Example:** group at AB=01, CD=any

- A is always 0 → write $\bar{A}$
- B is always 1 → write $B$
- C and D vary → omit
- **Result:** $\bar{A}B$

### 3.6 The XOR secret

**[HACK] Checkerboard pattern = XOR**

If the 1s and 0s alternate like a chessboard:

| | |
|---|---|
| 1 | 0 |
| 0 | 1 |

**Write directly:** $A \oplus B$ (XOR)

This case **cannot be simplified further**; don't waste time trying to group it.

### 3.7 Don't-care

**X can be treated as 1 or 0**

- If it makes a group bigger → treat as 1
- Otherwise → treat as 0 and ignore

Goal: make groups as large as possible.

## 4. RS Latch (Traffic-Light Rule)

**[HACK] NOR-gate RS - active-high**

**Traffic-light memory trick:**

- $S = 1 \to Q$ becomes 1 (Set)
- $R = 1 \to Q$ becomes 0 (Reset)
- Both 0 → **copy the previous state**
- Both 1 → write "forbidden" / "unstable"

**[HACK] NAND-gate RS - active-low**

**It is the opposite!**

- $\bar{S} = 0 \to Q$ becomes 1
- $\bar{R} = 0 \to Q$ becomes 0
- Both 1 → **copy the previous state**
- Both 0 → forbidden

### 4.1 RS truth table (must memorize)

| S | R | Q | What to write |
|---|---|---|---|
| 0 | 0 | Q | copy previous |
| 0 | 1 | 0 | write 0 |
| 1 | 0 | 1 | write 1 |
| 1 | 1 | ? | write "forbidden" |

### 4.2 How to solve waveform problems

**[HACK] Three steps for timing diagrams**

1. Draw vertical lines at every change point on S and R
2. For each time segment, read the values of S and R
3. Fill in Q using the truth table

**Mnemonic:** S high ⇒ Q high, R high ⇒ Q low, both low ⇒ copy!

## 5. Shannon Expansion (Copy-and-Paste Method)

**[HACK] Shannon expansion - just apply the formula**

**Do not derive it. Use the template:**

$$F = \bar{A} \cdot F_0 + A \cdot F_1$$

**Three steps:**

1. $F_0$: replace every $A$ with 0
2. $F_1$: replace every $A$ with 1
3. Plug into the formula

### 5.1 Shannon expansion example

**Example: $F = AB + \bar{A}C + BC$, expand w.r.t. A**

**Step 1: compute $F_0$ (A=0)**

Replace $A \to 0$ and $\bar{A} \to 1$:

$$\begin{aligned} F_0 &= (0)B + (1)C + BC \\ &= 0 + C + BC \\ &= C \quad \text{(absorption law)} \end{aligned}$$

**Step 2: compute $F_1$ (A=1)**

Replace $A \to 1$ and $\bar{A} \to 0$:

$$\begin{aligned} F_1 &= (1)B + (0)C + BC \\ &= B + 0 + BC \\ &= B \quad \text{(absorption law)} \end{aligned}$$

**Step 3: plug into the formula**

$$\boxed{F = \bar{A} \cdot C + A \cdot B}$$

**[HACK] Absorption law quick notes**

- $X + XY = X$ (keep the bigger term)
- $X + \bar{X}Y = X + Y$ (complement trick)

### 5.2 Two-variable Shannon expansion

**[HACK] Expand w.r.t. AB together**

**Formula:**

$$F = \bar{A}\bar{B}F_{00} + \bar{A}BF_{01} + A\bar{B}F_{10} + ABF_{11}$$

**Procedure:** substitute (A,B)=(0,0), (0,1), (1,0), (1,1).

### 5.3 Shannon expansion as a MUX

**Shannon = 2:1 MUX**

$F = \bar{A} \cdot F_0 + A \cdot F_1$ corresponds to:

- select = $A$
- input 0 = $F_0$
- input 1 = $F_1$

**[!] Common Shannon mistakes**

- When substituting $A = 0$, remember $\bar{A} = 1$ (don't swap them)
- Don't forget absorption when simplifying
- In a MUX, connect $I_0$ to $F_0$ and $I_1$ to $F_1$ (order matters)

## 6. Pipeline (Matching Game)

**[HACK] Match keywords to answers**

**When you see these keywords, write the corresponding answer immediately:**

| Problem says | Write |
|---|---|
| Data Hazard | Forwarding |
| Load-Use | Stall (bubble) |
| Branch/Jump | Flush |
| Structural | Add hardware |

### 6.1 5-stage pipeline (memorize)

**Five stages**

IF → ID → EX → MEM → WB

| | | |
|---|---|---|
| IF | Instruction Fetch | fetch instruction |
| ID | Instruction Decode | decode / read registers |
| EX | Execute | ALU operation |
| MEM | Memory | memory access |
| WB | Write Back | write back |

### 6.2 Pipeline timing diagram template

**[HACK] Copy this template**

```
    C1  C2  C3  C4  C5  C6  C7
I1  IF  ID  EX  MEM WB
I2      IF  ID  EX  MEM WB
I3          IF  ID  EX  MEM WB
I4              IF  ID  EX  MEM WB
```

**Pattern:** each instruction shifts one column to the right

### 6.3 Three hazard types

**Hazard categories**

**1. Structural hazard:**

- hardware resource conflict (e.g., single memory port)
- fix: add hardware

**2. Data hazard:**

- later instruction needs a value not produced yet
- fix: forwarding or stall

**3. Control hazard:**

- branch/jump fetches the wrong instruction
- fix: flush / prediction

### 6.4 RAW hazard

**[HACK] RAW detection**

**Look at registers:**

```
I1: ADD R1, R2, R3   ; writes R1
I2: SUB R4, R1, R5   ; reads R1 <- RAW!
```

**Test:** does a later instruction read a register that an earlier instruction writes?
If yes → RAW hazard.

### 6.5 Forwarding paths

**[HACK] Two forwarding routes**

**1. EX/MEM → EX:**
- forward ALU result from the previous instruction
- fixes 1-cycle RAW

**2. MEM/WB → EX:**
- forward result from two instructions back
- fixes 2-cycle RAW

### 6.6 Load-use must stall

**[!] Forwarding cannot fix load-use**

```
LW  R1, 0(R2)   ; data available only after MEM
ADD R3, R1, R4  ; needs data in EX
```

**You must insert 1 stall (bubble):**

```
 C1  C2  C3  C4  C5  C6
LW   IF  ID  EX  MEM WB
ADD      IF  ID  --  EX  MEM
```

"–" is the bubble/stall.

### 6.7 CPI calculation

**[HACK] CPI formula**

$$\text{CPI} = 1 + \text{stall rate}$$

**Example:** 30% of instructions are loads, and 50% of those cause a stall

$$\text{stall rate} = 0.3 \times 0.5 = 0.15$$
$$\text{CPI} = 1 + 0.15 = 1.15$$

### 6.8 Speedup

**Speedup formula**

$$\text{Speedup} = \frac{nk}{k+n-1} \to k$$

$k$ = number of stages, $n$ = number of instructions
For large $n$, speedup approaches $k$.

**[HACK] Pipeline problem recipe**

1. Draw the timing diagram
2. Find hazards (check registers)
3. Mark forwarding arrows or stalls
4. Compute CPI

## 7. Cheat Sheet (Read This Before the Exam)

### 7.1 Powers of two

| $2^4$ | $2^5$ | $2^6$ | $2^7$ | $2^8$ | $2^9$ | $2^{10}$ | $2^{16}$ |
|---|---|---|---|---|---|---|---|
| 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 65536 |

### 7.2 Value ranges

| Bits | Unsigned | Signed |
|---|---|---|
| 8 | 0∼255 | $-128 \sim +127$ |
| 10 | 0∼1023 | $-512 \sim +511$ |
| 16 | 0∼65535 | $-32768 \sim +32767$ |

### 7.3 Fast arithmetic notes

**Unsigned overflow:** result % $2^N$
**Signed overflow:** if result $> 2^{N-1} - 1$, subtract $2^N$
**Two's complement for negative:** $2^N - |X|$
**Two's complement to value:** MSB=1? value $-2^N$

### 7.4 K-Map positions

| | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 1 | 3 | 2 |
| 01 | 4 | 5 | 7 | 6 |
| 11 | 12 | 13 | 15 | 14 |
| 10 | 8 | 9 | 11 | 10 |

**Order:** 00-01-11-10

**Group sizes:** 1,2,4,8,16

**Corners can be grouped!**

### 7.5 RS latch

| S | R | Q |
|---|---|---|
| 0 | 0 | Hold |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | Forbidden |

**Mnemonic:** S high $\Rightarrow$ Q high, R high $\Rightarrow$ Q low

### 7.6 Shannon expansion

$$\boxed{F = \bar{A}F_0 + AF_1}$$

$F_0$: set $A = 0$, set $\bar{A} = 1$

$F_1$: set $A = 1$, set $\bar{A} = 0$

### 7.7 Pipeline matching

| | |
|---|---|
| Data Hazard | Forwarding |
| Load-Use | Stall |
| Branch | Flush |

**CPI** = 1 + stall rate

**5-stage:** IF-ID-EX-MEM-WB

### 7.8 Boolean algebra

| | |
|---|---|
| $\overline{A+B} = \bar{A}\bar{B}$ | De Morgan |
| $\overline{AB} = \bar{A} + \bar{B}$ | De Morgan |
| $A + AB = A$ | Absorption |
| $A + \bar{A}B = A + B$ | Complement trick |
| $A \oplus B = A\bar{B} + \bar{A}B$ | XOR |

**Good Luck!**

2026-01-13 10:00 | KN-A-310

*Stay calm. You've got this.*