ELEC1601 Introduction to Computer Systems

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING · USYD · EXAM REVISION SHEET

Full guide + ask Sia about ELEC1601

SIDE 1/2 DEFINITIONS · FORMULAS · DECISION RULES · MACHINE-STATE CHECKS

SEMESTER 2 · 2026

v2 · 2026-10-25

0 · Exam Map

COURSE.JSON · BRIEF

FQ 20 | XQ 10 | Lab 15 | EP 10 | XP | AP | FINAL 35

Permitted materials: not confirmed for this offering in the course materials we hold — check the exam instructions on the LMS.

ASSESSMENT	WT	CONTROL
Fundamentals Quizzes	20%	4 in-person · hurdle
Extension Quizzes	10%	3 in-person
Essential Lab Completion	15%	group + individual explanation · hurdle
Laboratory Attendance	0%	participation · hurdle
Essential / Extension / Advanced Project	10 / 5 / 5%	Essential is hurdle
Final exam	**35%**	2-hour written · AI prohibited · hurdle

The mandatory section uses multiple-choice questions based on in-semester tests; the advanced section carries a threshold for extra marks. **All essential assessments must be passed**, and lab participation is compulsory, to pass.

represent bits → trace state → connect hardware → test limits

Course-wide answer route: represent → trace → connect → test.

object before → operation or rule → state after → next control

Source-derived state control: every arrow must be a checkable transition.

1 · State-Trace Method

C1 · OVERVIEW

Program state is the complete set of variable, register, memory, flag and control-position values at one instant. Record only state that can be read, changed or depended upon.

STEP	WRITE	CHECK
Before	width/representation; PC/control point; registers; relevant memory; flags	preconditions explicit?
Execute	instruction/operation and operands	what is read; what is written?
After	new register, memory, flag and control state	next branch/call/iteration?
Test	ordinary · boundary · invalid	one rule isolated?

EXAM CHECK *Explain why the result follows. A correct output without the state transition is not a complete machine-level explanation.*

2 · Binary, Hex & Signed

C3 · GLOSSARY/FAQ

Positional binary: each bit contributes a power of two. ***Hexadecimal***: one digit 0–9/A–F compactly represents four bits. Preserve the fixed width while converting.

REPRESENTATION RULE

```
bit pattern + width + interpretation
→ machine meaning
negative two's-complement: invert
every bit, then add one
```

width meaning → hex ↔ binary same bits

group bits in fours → sign + overflow

Source-derived conversion route: never detach bits from width and representation.

QUESTION CUE	DO	SAY
binary ↔ hex	keep leading zeros; group into nibbles	same pattern, new notation
negative integer	write positive at stated width → invert → +1	two's-complement interpretation
same bits, two values	apply signed and unsigned interpretations separately	bits did not change
overflow	trace the fixed-width operation	state width before judgement

A bit pattern has no signed or unsigned meaning until the representation and width are specified.

For a full-credit conversion, label the source representation, keep leading zeros required by the width, show the intermediate pattern, then label the target notation and interpretation. To test overflow, inspect the result at the same fixed width; do not silently expand the representation or discard a carry without stating what the chosen width does.

fixed-width bits → unsigned? positional value / signed? two's complement

Same stored pattern; interpretation selects the value.

2b · Signed-Byte Check

COURSE WORKED EXAMPLE

```
+18 = 00010010
invert = 11101101
add 1 = 11101110 = 0xEE
signed: -18 · unsigned: 238
```

3 · Fixed vs Floating Point

C4 · GLOSSARY/FAQ

REPRESENTATION	STRUCTURE	WHAT TO CHECK
Fixed-point	integer pattern + agreed scale locating the radix point	resolution, rounding, representable limits
Floating-point	sign + exponent + significand	field meaning, precision spacing, range

FIXED-POINT MEANING

```
represented value = stored integer
pattern interpreted with the agreed
fixed scale
```

FLOATING-POINT READING ORDER

```
sign → exponent → significand →
represented magnitude
```

choose format → encode fields → check rounding → test range + compare

Source-derived encoding route: format choice controls later checks.

DECISION RULES

- If the task fixes a radix-point position, treat the pattern as **fixed-point** and keep the scale visible throughout.
- If the task separates sign, exponent and significand, decode those **floating-point fields** before comparing values.
- If the task asks "more range?", do not infer "more precision". These are different properties.
- If a decimal appears, allow for the fact that it may not have an exact finite binary encoding.

EXAM CHECK *More range does not guarantee more precision. State the representation consequence—rounding, overflow or comparison—not merely the field names.*

closer spacing → finer precision

wider endpoints → greater range

Precision concerns spacing; range concerns reachable magnitudes.

3b · Encoding Answer Frame

C4 · WORKED METHOD

1. Fix the value, width and chosen representation.
2. Write the scale or the sign/exponent/significand field roles.
3. Carry the encoding through without changing conventions.

4 · AVR Instruction Trace

C5 · GLOSSARY/FAQ

Mnemonic names the operation; ***register operands*** are the locations read or written; ***status flags*** record conditions such as zero, carry, negative or overflow.

mnemonic operation → operands read/write → result state → flags next flow

Source-derived AVR trace: operation → operands → result → flags/control.

TRACE COLUMN	QUESTION IT ANSWERS
PC / control	which instruction executes next?
mnemonic	what operation is named?
source operands	which registers/locations are read?
destination	which register/location changes?
flags	what arithmetic condition is now recorded?

An assembly sequence is correct only under the stated register contents, data width and calling assumptions.

5 · CPU & Memory Cycle

C6 · TOPIC/FAQ

program memory → fetch instruction → decode operands → execute datapath

state: regs · memory · flags · control

Source-derived architecture link: encoding reaches visible state through fetch/decode/execute.

OBJECT	ROLE	DO NOT CONFUSE
CPU datapath	moves/transforms data during execution	not the whole processor design
register file	fast operands, addresses, intermediate values	not general data memory
program memory	executable instructions	not mutable program data
data memory	mutable values	not instruction encoding
instruction encoding	supplies operation and operand information for fetch/decode	not the visible result
visible machine state	register, memory, flag and control changes after execution	not unstated timing detail

LIMIT *A simplified architecture diagram omits timing and implementation detail; use it to explain state, not as a complete processor design.*

6 · Addressing Modes

C7 · GLOSSARY/FAQ

Byte address identifies one byte-sized storage unit. ***Pointer register*** stores an address used for indirect access and may be updated.

pointer register holds → byte address selects → stored value is read/written

three different objects

Source-derived object map: pointer ≠ address ≠ value at that address.

MODE	LOCATION COMES FROM	TRACE
direct	location named explicitly	named address → value
indirect	pointer register	pointer value → address → value
indirect + update	pointer register, then modified	accessed location and pointer after access

ADDRESS TRACE

```
pointer before → effective byte
address → value read/written →
pointer after
```

Never write "the pointer contains the data" when it contains the address. A later access may change solely because the pointer changed.

7 · Runtime Stack

C8 · GLOSSARY/FAQ

stack pointer: top boundary

return information

saved state

local storage

activation record for one call

Source-derived stack model: one call's return information, saved state and locals.

EVENT	PROVE
push	value stored and stack pointer moves under the stated AVR convention
pop	value retrieved and matching pointer change
nested call	each activation record and return information remain distinct
return	caller state and continuation are recovered

Join the memory and stack traces: record the byte address touched, the value stored or retrieved, the stack pointer before and after, the activation record owning that slot, and the control position restored on return. If any one is missing, later nested-call state cannot be proved.

LIMIT *Stack growth direction and exact call behaviour are architecture-specific; follow the course's AVR convention.*

8 · Calls & Conventions

C9 · GLOSSARY/FAQ

caller sets arguments → call saves continuation → callee executes

preserve assigned state → return result + control

Source-derived call contract: arguments, continuation, preservation, result.

RESPONSIBILITY	QUESTION
parameter passing	which inputs or addresses cross the interface?
caller-saved	what must the caller preserve before calling?
callee-saved	what must the subroutine restore before returning?
result	where is the output and who may overwrite it?

A function can return the right sample result yet violate the calling convention and corrupt later execution.

9 · Branches & Loops

C10 · GLOSSARY/FAQ

compare / arithmetic → fresh flags → branch decision

body keeps invariant → progress to termination

Source-derived loop proof: relevant flags drive branch; invariant + progress establish correctness.

```
operation/comparison → condition flags
→ branch instruction → body update →
re-test
```

A ***loop invariant*** remains true before and after every iteration. Show it initially, show the body preserves it, then connect loop exit to the required result and boundary behaviour.

Interpret a branch only from flags produced by the immediately relevant comparison or arithmetic operation.

10 · Polling & Interrupts

C2 GLOSSARY/FAQ

IF THE CUE IS…	TRACE THIS	BOUNDARY
polling loop	read status → branch → repeat/service	software repeatedly checks
interrupt response	event → handler → shared state → resume	timing/handler duties controlled

REMEMBER *An interrupt improves responsiveness only when shared state, timing and handler responsibilities are controlled.*

ELEC1601

Self-Test Drills · Answer Paths · Classic Traps

INTRODUCTION TO COMPUTER SYSTEMS · USYD · EXAM REVISION SHEET

Full guide + ask Sia about ELEC1601

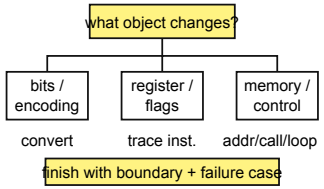
SIDE 2/2 FIVE WORKED DRILLS · IF-X-THEN-Y CARDS · FAST RECALL · EXAM-MORNING PLAN

SELF-TEST BEFORE READING ANSWERS

v2 · 2026-10-25

11 · Method Chooser

TOPIC MAPS · FAQ



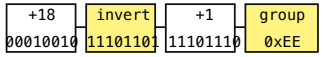
Source-derived chooser: identify the changing object before choosing a method.

IF YOU SEE…	DO THIS
bit pattern	state width + representation; convert without dropping bits
fixed/float value	state scale or fields; check precision/range
assembly sequence	trace operands, destination, flags and next control
pointer/address	separate pointer, address and stored value
call/return	track arguments, saved state, result, return information
branch/loop	find producing flags; show invariant and termination
input event	choose polling or interrupt trace; expose shared state/timing

12 · Worked 1 · Signed Byte

COURSE WORKED · 4 MARKS

Q. Represent −18 as an 8-bit two's-complement value, convert it to hexadecimal and explain why the same bits can mean 238.



same bits

signed −18 · unsigned 238

Complete number path from the source example; every intermediate pattern retained.

1. **1:** +18 = 00010010.
2. **1:** invert → 11101101; add one → 11101110.
3. **1:** group nibbles → 1110 1110 = 0xEE.
4. **1:** signed 8-bit meaning = −18; unsigned meaning = 238.

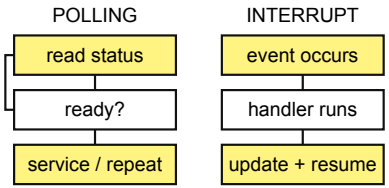
Self-mark: award nothing for a bare 0xEE if the width, inversion, addition and signed/unsigned interpretation are absent. Cover the answer, reproduce all three patterns, then explain in one sentence how identical bits support two meanings.

TRAP *Do not attach signed meaning until width and representation are explicit.*

13 · Worked 2 · Polling / Interrupt

C2 WORKED · 4 MARKS

Task. Trace when a microcontroller reads an input and changes an output under polling and interrupt designs.



Source-derived comparison: repeated software check versus event-triggered handler.

1. **1:** mark the complete program state before input service.
2. **1:** polling: show read, decision, repeat/service; interrupt: show event, handler entry and responsibilities.
3. **1:** trace shared-state and output changes, then the next control position.
4. **1:** qualify: improved responsiveness depends on controlled shared state, timing and handler duties.

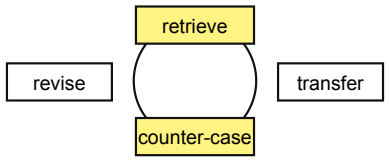
TRACE ROW	POLLING	INTERRUPT
trigger	software check	hardware event
between events	loop keeps checking	other work may run
proof burden	read/branch/repeat	entry/shared state/resume

13b · State-Trace Drill Card

C1 · C2

STEP	PC/CONTROL	REGISTERS	MEMORY	FLAGS/OUTPUT
before	____	____	____	____
read	____	____	____	____
decide	____	____	____	____
service	____	____	____	____
after	____	____	____	____

Before executing, predict the output for **ordinary, boundary and invalid** inputs. Change one condition at a time; otherwise the failed rule cannot be localised. For an interrupt trace, identify which state is shared, who may change it, what the handler must finish, and where normal execution resumes. For polling, show the repeated read and the exact decision that exits or services the loop.



Practice loop: retrieve → transfer → counter-case → revise.

FIX *Define program state before service: variables, registers, memory, flags and control position. In polling, name the repeated check; in interrupts, name handler entry, shared-state change and the resume point.*

14 · Worked 3 · AVR Trace

C5 WORKED · 4 MARKS

Task. Translate a small operation into AVR instructions and trace each register and flag change.

1. **1:** name the required operation and starting register contents.
2. **1:** state each mnemonic and which register operands it reads/writes.
3. **1:** after every instruction, record changed registers and status flags.
4. **1:** state the next control action; retain width and calling assumptions.

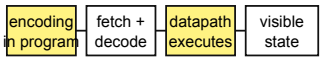
step	instruction	registers	flags/next
0	initial	given	given
1	mnemonic	old→new	flags
…	next	changes	next PC

Source-derived answer grid: never skip the initial state or flag-producing step.

TRAP *A familiar mnemonic does not prove correctness. Match its operation, operands, width, flags and calling assumptions to the task.*

14b · Architecture Drill

C6 WORKED



Source-derived path for connecting instruction encoding to machine state.

PROMPT	ANSWER MOVE
"where is instruction?"	program memory
"where are operands?"	register file and/or data memory, as stated
"what transforms them?"	CPU datapath during execute
"what can we observe?"	register, memory, flag and control changes

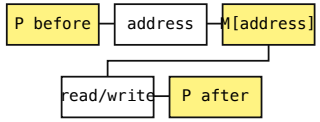
Boundary: the diagram is a reasoning aid; it omits timing and implementation detail.

WORKED CHECK *Start from given register contents and width. For each instruction, write its operation, operands read/written, resulting register state and new zero, carry, negative or overflow flag. Then state next control. If a row fails, repair the first unsupported transition; never infer later state from a familiar mnemonic.*

15 · Worked 4 · Address / Stack

C7–C8 WORKED · 4 MARKS

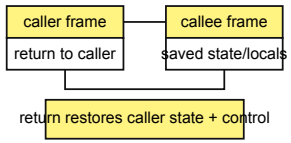
Task. Calculate an accessed location, show pointer updates, then trace nested-call stack state and return information.



record all five—even unchanged

Source-derived address calculation: pointer before and after bracket the access.

1. **1:** define the pointer register and the byte address it holds.
2. **1:** distinguish direct location naming from indirect pointer use.
3. **1:** state effective address, value read/written and pointer after.
4. **1:** qualify: pointer, address and stored value are different objects.



Source-derived nested-call check: frames and return information must not blur together.

```
SP before → call/push effect →
callee frame → nested effect →
return/pop effect → SP after
```

15b · Stack / Call Traps

C8–C9 FAQ

TEMPTING CLAIM	REPAIR
"stack always grows this way"	state the AVR convention used
"result correct, function correct"	also prove preserved state and return
"pop undoes any push"	match order, value and pointer state
"address equals contents"	write pointer → address → value

15c · Representation Traps

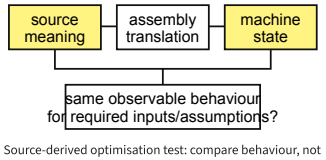
C3–C6 LIMITS

TRAP	REPAIR SENTENCE
bits "are negative"	At the stated width, under two's-complement interpretation…
hex changed the value	Hex is compact notation for the same grouped bits.
range = precision	Range controls magnitudes; precision controls spacing.
mnemonic alone proves result	Trace operands, destination and flags from stated contents.
diagram is complete CPU	It omits timing and implementation detail.

16 · Worked 5 · Branch / Compiler

C10–C11 WORKED · 4 MARKS

Task. Relate a structured loop or C-level construct to AVR assembly and justify an optimisation safely.



Source-derived optimisation test: compare behaviour, not surface instruction count.

1. **1:** define the source construct's required observable behaviour.
2. **1:** map operations and control structures to assembly instructions.
3. **1:** trace registers, memory, flags and branches through normal and boundary cases.
4. **1:** accept the change only if behaviour remains equivalent under the stated assumptions.

LOOP PROOF

1 · initial state establishes invariant
2 · body preserves invariant
3 · update makes progress toward exit
4 · exit + invariant implies required result

Source-derived loop proof: invariant and progress are separate obligations.

Shorter is not automatically correct or faster.

Different instruction sequences may be semantically equivalent; show the conditions under which they are.

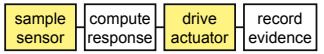
16b · Branch Judgement Cards

C10 FAQ

IF YOU SEE X	DO Y
branch mnemonic	name the flag condition and its producer
loop boundary	trace zero/one/final iteration behaviour
non-termination	inspect update, branch polarity and progress
optimised loop	compare observable state on same cases

16c · Sensor–Actuator Link

C12 GLOSSARY



Source-derived control cycle: observable action must connect back to bit- and instruction-level state.

17 · Classic Control Traps

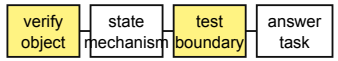
CHAPTER LIMITS

TRAP	REPAIR SENTENCE
pointer = value	The register holds an address; memory at that address holds the value.
sample output proves call	Also prove calling-convention preservation and return.
branch read in isolation	Use flags from the immediately relevant operation.
shorter = optimised	First prove equivalent observable behaviour.
interrupt always better	Only when shared state, timing and handler duties are controlled.

18 · Fast Recall

COURSE GLOSSARY

TERM	RECITE
AVR assembly	symbolic mnemonics encoding AVR operations
computer architecture	programmer-visible instruction set, registers, memory model, data paths
stack pointer	register identifying current top boundary of runtime stack
subroutine	reusable sequence entered by call, completed by return
activation record	one call's saved state, locals and return information
loop invariant	property true before and after every iteration
explainable evidence	trace, test and design records that justify system behaviour



Source-derived final review control: object → mechanism → boundary → requested conclusion.

19 · Exam-Morning Plan

COURSE STRATEGY · BRIEF

1. **Retrieve:** say the definitions of width/representation, state, pointer, stack pointer, flags, invariant and semantic equivalence without notes.
2. **Trace:** complete one blank table across bits → instruction → register/memory/flags → control flow.
3. **Transfer:** change one condition and predict the new state before checking.
4. **Counter-case:** test ordinary, boundary and invalid inputs; repair the first failed reasoning move.
5. **Verify operations:** check the current LMS instructions and exact timetable published by the Exams Office.

ANSWER RHYTHM

```
define object → state rule → show
transition → name visible
consequence → state boundary
```