MODULE P LANGUAGES AND IDES · P2

Assembly language and assemblers: mark scheme

20 marks · Cambridge 0478 syllabus 4.2.2 (assembly language is a form of low-level language that uses mnemonics; an assembler translates it into machine code), with 4.2.1 (advantages and disadvantages of low-level languages), 4.2.3 (compiler) and 3.1.4 (an instruction set's commands are machine code)

How to use. One mark per creditworthy point up to the maximum shown. **/** separates alternative words; **//** separates alternative answers. Words in (brackets) are not needed for the mark. Credit any other valid answer that shows the same understanding. Candidates do not need to know what any particular mnemonic does.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) 0478 4.2.1, 4.2.2	One mark for each correct row: A: High-level language B: Machine code C: Assembly language	**3**	A row with more than one tick scores 0.
1(b) 0478 4.2.2	One mark for the meaning, one mark for the example: A short code / abbreviation (of letters) that represents an instruction / operation / command (used in assembly language instead of binary) LDA // ADD // STO	**2**	Accept "a word that is easy to remember that stands for a machine code instruction". Accept a whole line from program C (e.g. "LDA 6"). Do not accept a number alone (6, 7, 8), a binary pattern from program B, or a word from program A (e.g. if, then).
2(a) 0478 4.2.1, 4.2.2, 4.2.3	One mark for each correct row: Form of low-level language: True A compiler translates assembly language: False Allows direct manipulation of the hardware: True Will run on any other type of processor without being changed: False	**4**	A row with both boxes ticked scores 0.
3(a) 0478 4.2.2, 3.1.4	Any two from: Assembly language uses mnemonics // (text) codes, but machine code is (written in) binary // 1s and 0s Assembly language must be translated (by an assembler) before it can be executed, but machine code can be executed directly by the CPU // without translation Machine code is the CPU's instruction set // the commands the CPU processes, but assembly language represents these commands (using mnemonics) Assembly language is easier (for a person) to read / write / understand / debug than machine code	**2**	max 2. Each point must make the difference clear: both sides stated or clearly implied (e.g. "assembly uses mnemonics instead of binary"). Do not accept "one is high-level and one is low-level" (both are low-level).

QUESTION	ANSWER	MARKS	GUIDANCE
4(a) 0478 4.2.2, 3.1.4	One mark for: An assembler (is the translator needed) Any two from: The CPU can only execute // process machine code // binary instructions The CPU's instruction set is (made of) machine code commands Assembly language uses mnemonics // is not binary, so the CPU cannot execute it directly The assembler converts the mnemonics // assembly language (instructions) into machine code // binary	**3**	max 3. Do not award the translator mark for compiler or interpreter. Do not accept "the computer cannot read it" alone (too vague). Accept "translates it so the CPU can understand it" only with machine code // binary.
5(a) 0478 4.2.1, 4.2.2	One mark for each requirement linked to a feature of assembly language, one mark for the matching expansion (max 2 requirements): **Small memory:** assembly language // low-level code can be memory efficient // produce a small program ... ... because the programmer writes each instruction for the processor // controls exactly which instructions // memory locations are used, so there are no unnecessary instructions **Control the hardware directly:** assembly language allows direct manipulation of the hardware ... ... because its instructions are (mnemonics for) the processor's own machine code // instruction set, so the driver can access specific memory locations // registers // the print head directly **Respond quickly:** assembly language code can execute quickly // efficiently ... ... because it can be written specifically for that processor // uses the processor's own instructions, with no unnecessary instructions	**4**	max 4: max 2 per requirement, max 2 requirements. The expansion mark needs a matching feature mark. Credit any other valid expansion that explains the link. "It runs faster" earns only the feature mark for "respond quickly"; the expansion needs a reason. Do not accept "it does not need a translator" (it needs an assembler).
5(b) 0478 4.2.1	Any two from: Harder to read // write // understand (than a high-level language) Harder to debug // errors are harder to find and fix Machine dependent // only runs on one type of processor // must be rewritten for a different type of processor Takes longer to write // more lines of code are needed (for the same task) The engineer must know the processor's instruction set // hardware in detail	**2**	max 2. Do not accept "it runs slower", "it needs a compiler" or "it uses more memory" (all wrong for assembly language). Do not accept "harder to use" alone (too vague).

Total: 20 marks