**Opracowane pytania**

**Z1. Użycie języka assemblera do programowania można rekomendować, gdy program jest:**

1. ogólnego przeznaczenia na jedną platformę komputerową
2. ogólnego przeznaczenia na wiele platform komputerowych
3. **systemem wbudowanym na jedno urządzenie**
4. **sterownikiem urządzenia komputera**
5. **procedurą obsługi przerwania lub wyjątku**
6. systemem operacyjnym komputera

**Z2. Wskaż skutki użycia dyrektywy REPEAT (w języku asemblera) w następującej konstrukcji:**

REPEAT wyrażenie

/instrukcja

ENDM

1. wykonanie “instrukcja” tyle razy ile wskazuje wyrażenie
2. **utworzenie kodu “instrukcja” tyle razy ile wskazuje wyrażenie**
3. wykonywanie “instrukcja” tak długo, aż wyrażenie uzyska wartość true
4. **wydłużenie tekstu programu źródłowego w stosunku do wersji program bez użycia dyrektywy REPEAT**
5. skrócenie czasu wykonania programu w stosunku do wersji program bez użycia dyrektywy REPEAT
6. utworzenie kodu “instrukcja” tak długo aż wyrażenie uzyska wartość TRUE

**Z3. Makrodefinicja ma następującą postać:**

MA MACRO X1,X2

MOV AX,X2

MOV X1,AX

ENDM

Jak będzie wyglądać rozwinięcie makrowywołania: MA D1, D2

1. **MOV X2,D2**  
   MOV AX,X2  
   MOV X1,AX  
   MOV D1,X1
2. **MACRO D1,D2**  
   MOV AX,X2  
   MOV X1,AX  
   ENDM
3. **MOV AX,X2**  
   MOV X1,AX
4. **MOV AX,D1**  
   MOV D2,AX
5. **MOV AX,D2**  
   MOV D1,AX  
   ENDM
6. **MOV AX,D2  
   MOV D1,AX**

**Z4. Wskaż zdanie prawdziwe dotyczące dyrektyw asemblacji warunkowej:**

1. Użycie dyrektyw asemblacji warunkowej umożliwia wykonanie bloków programu w zależności od warunków sprawdzanych przez program.
2. **Wybór bloków programu źródłowego, które zostaną przetłumaczone na ML, zależy od kombinacji warunków określonych dyrektywami asemblacji warunkowej**
3. Blok programu źródłowego objęty dyrektywami asemblacji warunkowej będzie tłumaczony w zależności od warunków sprawdzanych przez program
4. Asemblacja warunkowa wykorzystuje instrukcje warunkowe ML w celu umożliwienia wariantowości programu
5. **Wybór bloków programu źródłowego, które zostaną przetłumaczone na ML następuje w czasie asemblacji programu źródłowego**
6. Jeżeli w dyrektywach asemblacji warunkowej zastąpić stałe zmiennymi, to bloki programu wyznaczone tymi dyrektywami będą wykonywane warunkowo, w zależności od stanów warunków wyznaczonych w czasie działania programu.

**Z5. Wskaż zdania prawdziwe dotyczące makroinstrukcji.**

1. **Makroinstrukcja jest nazwanym ciągiem instrukcji ML, który zostanie wstawiony przez asembler do programu źródłowego, ilekroć asembler napotka nazwę tej makroinstrukcji**
2. Użycie Makroinstrukcji daje identyczny skutek jak użycie procedury //dopytać ZBYSZKA jak rozumie SKUTEK(nie trzeba dopytywać napewno to adp nie pasuje bo przecież w macro nie ma angażowania stosu)<-(ale rezultat bedzie taki sam)
3. Użycie Makroinstrukcji jest mniej wygodne niż użycie procedury, gdyż w przypadku makroinstrukcji nie można przekazywać parametrów
4. Makroinstrukcje służą do utworzenia modułów
5. **Używanie makroinstrukcji może przyspieszyć działanie programu, w stosunku do wersji używającej procedur w miejsce makroinstrukcji**
6. **Używanie makroinstrukcji może wydłużyć kod źródłowy programu, w stosunku do wersji używającej procedur w miejsce makroinstrukcji**

**Z6. Zaznacz zdania prawdziwe dotyczące danej utworzonej w języku asemblera dyrektywą generacji danej użytą w konstrukcji:**

**SYMBOL: DB <wyrażenie>**

1. SYMBOL uzyska wartość <wyrażenie>
2. Obraz komórki pamięci , której lokalizacja oznaczona została przez SYMBOL, uzyska wartość <wyrażenie>
3. **program wpisze do pamięci wartości <wyrażenia>**
4. **Wartością SYMBOL jest adres komórki w pamięci, gdzie zostanie umieszczona wartość <wyrażenie>**
5. Zmiana w czasie wykonania programu wartości zmiennych występujących w <wyrażenie> zmieni wartość danej generowanej dyrektywą
6. <wyrażenie> musi być wyliczane w czasie asemblacji programu

**Z7. Rozgałęzienie warunkowe od wyniku porównania a>=b (a,b bez znaku) powinno wykorzystać flagę(i) w następujący sposób:**

1. Skok, gdy flaga przepełnienia ustawiona
2. Skok, gdy flaga przeniesienia ustawiona
3. Skok, gdy flaga zera ustawiona
4. Skok, gdy flaga przepełnienia wyzerowana i flaga zera ustawiona
5. **Skok, gdy flaga przeniesienia wyzerowana**
6. Skok, gdy flaga znaku wyzerowania lub flaga zera ustawiona

**Z8. Kiedy program wykonujący obliczanie arytmetyczne należy zakończyć rozgałęzieniem do obsługi błędu obliczania wykonywanym, gdy flaga przepełnienia jest ustawiona?**:

1. Gdy obliczanie polega na sumowaniu liczb bez znaku
2. Jeżeli a<b, a obliczanie polega na a-b
3. **Gdy obliczanie polega na odejmowaniu liczb ze znakiem**
4. **Gdy obliczanie polega na sumowaniu liczb ze znakiem**
5. Gdy obliczanie polega na odejmowaniu liczb bez znaku
6. **Jeżeli wynik operacji przekroczy zakres określony liczbą bitów reprezentacji liczby**

**Z9.Jeżeli, w odniesieniu do parametru przekazywanego pomiędzy programem a procedurą: “procedura nie modyfikuje zmiennej odpowiadającej temu parametrowi”, to:**

1. Taki parametr określamy jako “wejściowo-wyjściowy”
2. Taki parametr nie może być przekazany poprzez zmienną
3. Taki parametr określamy jako “wyjściowy”
4. Taki parametr określamy jako “stałą programu”
5. **Taki parametr nie może być przekazany poprzez odwołanie (referencję)**
6. **Taki parametr określamy jako “wejściowy”**

**Z10. Wskaż zdania poprawne dotyczące konstrukcji pętli WHILE**

1. Konstrukcja zawiera: skok warunkowy poza pętle; na końcu pętli sprawdzanie warunku i skok bezwarunkowy na początek
2. **Konstrukcja zawiera: sprawdzanie warunku, skok warunkowy poza pętle; na końcu pętli skok bezwarunkowy na początek**
3. Konstrukcja zawiera: sprawdzanie warunku, skok bezwarunkowy poza pętle; na końcu pętli skok warunkowy na początek
4. Instrukcja skoku warunkowego musi być poprzedzona etykietą
5. **Instrukcja sprawdzania warunku musi być poprzedzona etykietą**
6. Pierwsza instrukcja po pętli nie wymaga etykiety

**Z11. Które ze zdań odnoszących się do przekazywania parametru w “kodzie programu” nie są prawdziwe:**

1. Możliwe jest przekazywanie zmiennych o dowolnie zmiennej długości
2. Jeżeli kod programu umieszczony jest w segmencie bez prawa zapisu, wówczas to samo wywołanie procedury nie może się odbyć z każdorazowo inną wartością parametru
3. Parametr umieszcza się bezpośrednio po instrukcji wywołania procedury
4. **Nie wolno w procedurze modyfikować zapisanego na stosie adresu powrotu**
5. Parametr można umieścić bezpośrednio po dowolnej instrukcji programu
6. **W każdej architekturze procesora parametry przekazywane tą drogą są tylko stałymi**

**Z12.Zaznacz prawdziwe zdania odnoszące się do wykonania działania A+C, gdzie Ah jest starszą a Al młodszą częścią liczby A; Ah,Al,C mają długość 8 bitów; procesor wykonuje operacje arytmetyczne wyłącznie na danych 16 bitowych**

1. A+C oblicza się pojedynczą instrukcją procesora
2. Należy dodać C do Al, a następnie do Ah
3. **Należy rozszerzyć C do formatu A i dodać pojedynczą instrukcją**
4. Należy utworzyć 16 bitowy odpowiednik C, w którym starszy bajt zawiera same zera
5. Należy utworzyć 16 bitowy odpowiednik C, w którym starszy bajt zawiera same jedynki
6. Należy dodać C do Al, a następnie dodać 0 do Ah z uwzględnieniem przeniesienia z dodawania C do Al

**Z13.Opisz w krokach jak zaimplementować pętle WHILE w procesorze, w którym wszystkie rozkazy wykonywane są warunkowo (ARM)**

**; Display integers 1 – 10:**

**mov eax,0**

**.WHILE eax < 10**

**inc eax**

**call WriteDec**

**call Crlf**

**.ENDW**

**mov eax,0   
@WHILE:**

**//warunek**

**jae @C0001 ; fałsz? Opuść pętlę**

**//instrukcja**

**jnb @WHILE**

**@C0001:**

**Z13.** **Opisz w krokach, jak zaimplementować przesunięcie o 1 pozycję w lewo słowa 16-to bitowego w procesorze, w którym występują wyłącznie 8-bitowe przesunięcia**

**shl al, 1**

**rcl ah, 1**

**Z1 Konsolidacja polega na:**

1. **Scaleniu różnych plików źródłowych w jeden plik wynikowy.**
2. Załadowaniu różnych modułów we wspólny obszar pamięci.
3. Przypisaniu wartości symbolom zdefiniowanym poza bieżącym modułem.
4. Zapisaniu w pliku wynikowym tablicy symboli eksportowanych.

**Z2. Wskaż zdanie NIEprawdziwe dotyczące deklarowania procedury dyrektywną „PROC”**

1. Użycie „PROC” przy etykietce wejściowej procedury służy wskazaniu, od którego miejsca w programie rozciąga się lokalność zmiennych zadeklarowanych w procedurze.
2. **użycie „PROC” z następującą po niej listą nazw parametrów spowoduje, że parametry z listy zostaną zmiennymi lokalnymi(wewn.) i zostanie utworzona odpowiednia ramka na stosie**
3. użycie „PROC” z następującą po niej listą parametrów umożliwia zadeklarowanie parametrów przekazywanych między programem wywołującym, a procedurą i określenie ich typów
4. dyrektywa „PROC” powinna być poprzedzona etykietą (nazwą) procedury

**Z3.a Które ze zdań odnoszących się do przekazywania parametru w „kodzie programu” nie jest prawdziwe:**

1. Możliwe jest przekazywanie zmiennych o zmiennej długości
2. **Jeżeli kod programu umieszczony jest w segmencie bez prawa zapisu, wówczas to samo wywołanie procedury nie może się odbyć z każdorazowo inną wartością parametru**
3. Parametr umieszcza się bezpośrednio po instrukcji wywołania procedury
4. **nie wolno w procedurze modyfikować adresu powrotu, który został zapisany na stosie w wyniku wywołania procedury**

**Z4. Wskaż prawdziwe zdanie dotyczące deklarowania ciała procedury dyrektywą taką jak „ENDP”**

1. Musi wystąpić na końcu każdego rozgałęzienia procedury
2. zastępuje instrukcję RET
3. Umożliwia asemblerowi określić zakres lokalności wewnętrznych zmiennych procedury
4. **Stosowana jednocześnie z dyrektywą PROC umożliwia asemblerowi określić zakres lokalności wewn. zmiennych procedury**

**Z5. Relokacji wymagają:**

1. wszystkie instrukcje maszynowe
2. instrukcje z adresowaniem natychmiastowym
3. instrukcje z adresowaniem względnym
4. **wszystkie instrukcje z wyjątkiem instrukcji określonych w pkt B i C powyżej**

**Z6. Relokacja kodu programu podczas ładowania programu może polegać na:**

1. zastąpieniu dotychczasowych kodów rozkazów nowymi kodami rozkazów wynikających z bieżącego rozmieszczenia programu w pąmięci
2. dodaniu adresu położenia początku programu do adresów odwołań polach argumentów tych rozkazów, które zostaną rozmieszczone w innych lokalizacjach, niż lokalizacje ustalone podczas asemblacji programu
3. **Dodaniu adresu położenia początku programu do adresów odwołań w polach argumentów tych rozkazów, które odwołują się do lokalizacji zmienionych w stosunku do położenia ustalonego podczas asemblacji programu**
4. nadanie nowych wartości adresów etykietom występującym w programie, poprzez dodanie adresu położenia początku programu

**Z7. Wskaż zdanie prawdziwe dotyczące traktowania wyrażeń dla potrzeb relokacji:**

1. różnica wyrażania względnego i bezwzględnego ma wartość bezwzględną (wyrażenie nierelokowalne)
2. różnica wyrażeń bezwzględnych ma wartość bezwzględną (wyrażenie nierelokowalne)
3. różnica wyrażeń względnych ma wartość względną (wyrażenie relokowalne)
4. **różnica wyrażeń względnych ma wartość bezwzględną (wyrażenie nierelokowalne)**

**Z8.Jeżeli parametry są przekazywane do procedury poprzez stos, wówczas prawdziwe jest, że:**

1. **W celu przekazania parametrów należy zarezerwować na stosie „ramkę” na odpowiednią liczbę bajtów i wywołać procedurę**
2. należy opróżnić stos z parametrów bądź w procedurze, bądź na wyjściu z procedury w programie głównym
3. w ciele procedury pierwszą operacją powinno być odczytanie pierwszego parametru ze stosu instrukcją POP (pobierz ze stosu)
4. w ciele procedury możliwe jest odczytywanie parametrów bez usuwania ich ze stosu

**Z9. Jeżeli w programie znajduje się dyrektywa taka, jak INVOKE: z nazwą procedury i następującą po niej listą parametrów, to:**

1. **makroasembler umieści w kodzie programu instrukcje zapisania parametrów na stos i wywołania procedury**
2. makroasembler umieści w kodzie programu instrukcję wywołania procedury, a w dalszej kolejności instrukcja tworzące ramkę na stosie dla parametrów lokalnych
3. makroasembler nie utworzy żadnych instrukcji poza instrukcją „wywołania procedury”, natomiast sprawdzi zgodność typów parametrów z zadeklarowanymi dyrektywą prototypu procedury („PROTO”) i w przypadku niezgodności zasygnalizuje błąd
4. **makroasembler zastąpi instrukcje odwołujące się do symboli przekazywanych parametrów instrukcjami odwołującymi się do lokalizacji parametrów na stosie**

**Z10. Jeżeli procedura przekazuje wyniki na stosie, to:**

1. żadne z poniższych
2. wykonuje instrukcję RET, a następnie zapisuje na stos wyniki tak, aby były dostępne w programie wywołującym
3. zapisuje wyniki na stos, następnie wskaźnik do lokalizacji wyników na stosie wpisuje do wybranego rejestru, ustawia wskaźnik stosu tak, aby wskazywał adres powrotu wykonuje instrukcję RET
4. zapisuje na stos wyniki, a następnie wykonuje instrukcję RET

**Z11. Które ze zdań odnoszących się do implementacji zmiennych lokalnych (procesory Intel 286+) jest prawdziwe:**

1. użycie dyrektywy „LOCAL” z listą nazw parametrów jest równoważne użyciu instrukcji ENTER z parametrem całkowitym
2. użycie dyrektywy „LOCAL” z listą nazw parametrów jest równoważne użyciu instrukcji LEAVE
3. **użycie dyrektywy „LOCAL” z listą nazw paramerów jest równoważne użyciu pary instrukcji :ENTER (po wejściu do procedury) i LEAVE (przed wyjściem z procedury)**
4. działaniu dyrektywy „LOCAL” nie można przypisać żadnego równoważnego kodu ASM

**Z12.Wskaż właściwą kolejność działań wykonywanych, gdy biblioteka systemowa jest ładowana dynamicznie przez system operacyjny**

**X- procedura biblioteczna, Y-loader dynamiczny, Z- wywołanie systemowe, W-obsługa wywołania, Q-program główny**

1. X-Y-Z-W-Q
2. Q-Y-W-X-Q
3. Q-X-Q
4. **Q-Z-W-Y-X-Y-Z-Q**

**Z13. Które ze zdań odnoszących się do dynamicznego tworzenia zmiennych lokalnych procedury, NIE jest prawdziwe:**

1. tą samą procedurę, z dynamicznie utworzonymi zmiennymi lokalnymi mogą wywoływać różne równoległe wykonywane programy
2. aby dynamicznie utworzyć zmienne procedury, należy zarezerwować odpowiednią ilość miejsca na stosie i wywołać procedurę
3. procedurę, z tak utworzonymi zmiennymi lokalnymi, można wywoływać rekurencyjnie
4. **W czasie działania procedury nadrzędnej dynamiczne zmienne lokalne nie mają lokalizacji w pamięci, dlatego nie można się do nich odwołać z procedury nadrzędnej**

**Z14. Co zawiera Program Linking Table (PLT) w pliku wynikowym w formacie ELF?**

1. **kody skoków z adresowaniem pośrednim pamięciowym do dynamicznie zmiennych adresów**
2. adresy ustalone podczas konsolidacji dynamicznej
3. tablice symboli importowanych
4. tablicę symboli eksportowanych

**Z15. Jaką postać powinny mieć moduły programu, aby mogły zostać poddane konsolidacji statycznej?**

1. programu źródłowego (plik tekstowy)
2. **relokowalnego pliku wynikowego z tablicą symboli importowanych i eksportowanych**
3. pliku (wynikowego) ładowalnego
4. relokowalnego pliku wynikowego z tablicą symboli importowanych

**Z16. Jaką informacją musi dysponować assembler, żeby przetłumaczyć do postaci konsolidowanej moduł zawierający odwołania do symboli importowanych**

1. wartościami symboli
2. adresami odwołań do symboli
3. **formatem symboli**
4. identyfikacją modułu, w którym symbol zdefiniowano

**Z17. Wskaż zdanie prawdziwe dotyczące konsolidacji programów**

1. importowany punkt wejścia jest relokowalny
2. eksportowany punkt wejścia jest relokowalny
3. eksportowany symbol jest relokowalny
4. **importowany symbol jest relokowalny**

**Z18. Rekordy modyfikacji w pliku konsolidowanym zawierają:**

1. identyfikację sposobu relokacji symboli
2. identyfikację operacji, którą ma wykonać z użyciem symbolu eksportowanego
3. instrukcję w której należy umieścić symbol importowany
4. **identyfikację operacji, którą ma wykonać z użyciem symbolu importowanego**