Zebrane z: Kolos 1.doc , PPS\_I\_KOLOKOWIUM.doc, zal zaoczni.docx, Opracowane pytania-PS.docx

**1. Jaki jest skutek użycia dyrektywy REPEAT (w języku asemblera) w następującej konstrukcji:**

REPEAT wyrażenie

instrukcja

ENDM

1. Wykonanie „instrukcja” tyle razy, ile wskazuje wyrażenie.
2. **Umieszczenie kodu „instrukcja” tyle razy, ile wskazuje wyrażenie.**
3. Wykonywanie „instrukcja” tak długo, aż wyrażenie uzyska wartość TRUE.
4. Utworzenie kodu „instrukcja” tak długo, aż wyrażenie uzyska wartość TRUE.

**2. Jaki jest skutek użycia dyrektywy REPEAT (w języku asemblera) w następującej konstrukcji:**REPEAT wyrażenie   
Instrukcja   
ENDM   
a) Wykonanie „instrukcji” tyle razy ile wskazuje wyraźnie   
**b) Utworzenie kodu „instrukcji” tyle razy ile wskazuje wyrażenie**   
c) Wykonanie instrukcji tak długo aż wyraźnie uzyska wartość TRUE   
d) Utworzenie kodu instrukcji tak długo aż wyraźnie uzyska wartość TRUE

**3. 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

**4. Makrodefinicja ma następującą postać:**MA MACRO X1,X2   
MOV AX,X2   
MOV X1,AX   
ENDM   
Jak będzie wyglądało rozwinięcie makrowywołania MA D1,D2   
a) MOV AX,D2   
MOV D1,AX   
ENDM   
**b) MOV AX,D2   
MOV D1,AX**c) MOV AX,X2   
MOV X1,AX   
d) MOV AX,D1   
MOV D2,AX

**5. 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**

**6. Wskaż, które elementy pliku wynikowego potrzebne są do debuggingu:**

1. Lista symboli lokalnych
2. Lista pozycji relokowalnych
3. Rozmiar kodu
4. Nazwa pliku

**7. Wskaz zdanie prawdziwe   
a) Jedna instrukcja ASM przypisana jest dokładnie jednej instrukcji ML**b) W ogólności jedna instrukcja ASM może być przetłumaczona na grupę instrukcji ML   
c) W ogólności jest możliwe ze grupa instrukcji ASM może być przetłumaczona na jedna instrukcje ML   
d) Żadne z powyższych nie jest prawdziwe

**8. Wskaż zdanie prawdziwe:**

**a) jednej instrukcji ASM przypisana jest jedna instrukcja ML**

b) w ogólności jedna konstrukcja ASM może być tłumaczona na grupę instrukcji ML

c) w ogólności możliwe jest że grupa instrukcji ASM może być przetłumaczona na jedną instrukcję ML

d) zdanie z punktu B jest prawdziwe tylko w odniesieniu do dyrektyw asemblera

**9. Jaką funkcję w asemblerze pełni dyrektywa dołączenia dodatkowego(-ch) pliku(-ów) do pliku źródłowego (np. INCLUDE):**

1. Sterowanie segmentami
2. Sterowanie loaderem (programem ładującym).
3. Sterowanie programem źródłowym.
4. Konsolidacja z podprogramami.

**10. Dyrektywa generacji danej (np. DW, DB, DD itp.) powoduje:**

1. Nadanie wartości symbolowi w celu zastąpienia nią tego symbolu przez asembler.
2. Nadanie wartości obrazowi komórki(-ek) pamięci, której lokalizacja oznaczona została symbolem.
3. Wpisanie do pamięci wartości podanej dyrektywą.
4. Żadna z powyższych.

**11. Kolejność czynności asemblacji:**

(a) określenie liczby bajtów zajmowanych przez każdą instrukcję

(b) przypisanie instrukcjom kodów (w tym wartości argumentów)

(c) analiza leksykalna

(d) analiza składniowa

1. a,b,c,d
2. c,d,a,b
3. c,d,b,a
4. b,c,a,d

**12. Zaznacz zdanie prawdziwe dotyczące tablicy symboli w asemblerze:**

1. Dwuprzejściowy tworzy tablicę symboli w drugim przejściu
2. W tablicy symboli każdemu symbolowi przyporządkowany odpowiadający mu adres.
3. Jednoprzejściowy nie s.. tablicy symboli
4. **Dla każdego symbolu tablica symboli zawiera także liczbę bajtów, jaką zajmuje wartość symbolu w ML.**

**13. Zaznacz zdanie prawdziwe dla zagadnienia odwołania w przód:**

1. **W asemblerze jednoprzejściowym nie mogą występować odwołania w przód.**
2. Jeżeli asembler jednoprzejściowy używa wablicy symboli, to nie występuje w nim problem odwołań w przód.
3. W asemblerze dwuprzejściowym nie mogą występować odwołania w przód.
4. Nie można przypisać wartości do symbolu, jeżeli ten symbol został w programie zdefiniowany później niż występuje odwołanie do niego.

**14. 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

**15. Relokacja kodu programu podczas ładowania programu może polegać na**a) Zastąpieniu dotychczasowych rozkazów nowymi rozkazami wynikającymi z bieżącego rozmieszczenia programu w pamięci   
**b) Dodaniu adresu początkowego położenia programu do adresów odwołań w tych rozkazach, które zostaną rozmieszczone w innych lokalizacjach niż lokalizacje ustalone podczas asemblacji programu**c) Dodaniu adresu początkowego położenia programu do adresów odwołań w tych rozkazach które odwołują się do lokalizacji zmienionych w stosunku do położenia ustalonego podczas asemblacji programu   
d) Żadne z powyższych 

**16. Wskaż zdanie prawdziwe dotyczące wyrażeń:**

* 1. Różnice wyrażenia względnego i absolutnego ma wartość absolutną.
  2. Suma wyrażeń względnych ma wartość absolutną.
  3. Różnica wyrażeń względnych ma wartość względną.
  4. Różnica wyrażeń względnych ma wartość absolutną.

**17. 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)**

**18. Wskaż zdanie prawdziwe dotyczące wyrażeń:**

a) wyr. względne - wyr. absolutne = wartość absolutna

b) wyr. względne + wyr. względne = wartość absolutna

c) wyr. względne - wyr. względne = wartość względna

**d) wyr. Względne + wyr. Względne = wartość względna**

**19. 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

**20. Użycie języka asemblera do programowania można rekomendować gdy program jest:**a) Ogólnego przeznaczenia na jedna platformę komputerowa   
b) Ogólnego przeznaczenia na wiele platform komputerowych   
**c) Systemem wbudowanym na jedno urządzenie**d) Żadne z powyższych

**21. W procesorach z segmentowaną przestrzenią adresową:**

* 1. Procesor posiada specjalną instrukcję do wywołania procedury zlokalizowanej poza bieżącym segmentem, natomiast instrukcja procedur jest taka sama jak dla procedur zlokalizowanych w bieżącym segmencie.
  2. Wywołanie procedur umieszczonych poza bieżącym segmentem programu jeśli możliwe i nie ma konieczności . wywołań w ramach bieżącego segmentu i poza tym segmentem.
  3. Procesor posiada instrukcje do wywołania i powrotu z procedury zlokalizowanej poza bieżącym segmentem, inne od instrukcji dla procedur zlokalizowanych w bieżącym segmencie.
  4. Nie jest możliwe wywołanie procedur umieszczonych poza bieżącym segmentem programu.

**22. W procesorach z segmentowaną przestrzenią adresowania:**a) Nie jest możliwe zagnieżdżenie procedur   
b) Zagnieżdżanie procedur jest zawsze możliwe, niezależnie czy procedura zagnieżdżona mieści się całkowicie w segmencie kodu   
**c) Zagnieżdżanie procedur jest możliwe pod warunkiem ze procedura zagnieżdżona mieści się całkowicie w bieżącym lub innym segmencie kodu**  
d) Żadne z powyższych nie jest prawdziwe

**23. Które ze zdań odnoszących się do przekazywania parametrów w rejestrach jest** **nieprawdziwe:**

1. Nie jest możliwe przekazywanie w ten sposób parametrów „wyjściowych”.
2. Liczba przekazywanych parametrów jest ograniczona.
3. Czas dostępu do parametru jest najkrótsza z możliwych.
4. Przekazywanie parametrów do procedury polega na załadowaniu ich wartości do odpowiednich rejestrów w wywołaniu procedury.

**24. 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

**F) W każdej architekturze procesora parametry przekazywane tą drogą są tylko stałymi**

**25. 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**

**26. Które ze zdań odnoszących się do przekazywania parametrów w komórkach pamięci jest prawdziwe:**a) nie jest możliwe przekazywanie w ten sposób parametrów „wejściowych”/”wyjściowych”   
**b) liczba przekazywanych parametrów jest ograniczona**c) czas dostępu do parametrów jest najkrótszy z możliwych   
d) nie jest możliwa rekurencyjne wywołanie tej procedury

**27. Jeżeli procedura przekazuje wyniki na stosie to**:   
a) Żadne z poniższychb) Wykonuje instrukcję RET, a następnie zapisuje na stosie wyniki tak, aby były dostępne w programie wywołującym   
**c) Ustawia wskaźniki stosu tak, aby wskazywały adres powrotu, następnie wskaźn. do lokalizacji wyników na stosie wpisuje do wybranego rejestru i wykonuje instr. RET**d) Zapisuje na stosie wyniki, a następnie wykonuje instr. RET 

**28. 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 błędów i wywołać procedurę.**
2. Należy opróżnić stos z parametrów bądź w procedurze bądź po wyjściu z procedury w programie głównym.
3. Parametry należy odczytać ze stosu instrukcją POP (pobierz ze stosu).
4. Ramka na stosie w całości składa się z przekazywanych parametrów.

**29. Gdy następuje przekazanie parametru ‘poprzez nazwę’ to do procedury jest przekazywany**a) Wskaźnik do zmiennej (adres)   
b) Wskaźnik do zmiennej, wartość zmiennej kopiowana do zmiennej wewnętrznej   
c) Parametr, procedura nie zmienia jego wartości gdyż kopiuje jego warto wewnątrz procedury(?)   
**d) Wskaźnik do funkcji obliczającej adres zmiennej**

**30. Jeżeli parametry są przekazywane do procedury poprzez adres wówczas prawdziwe jest ze:**a) należy zarezerwować na stosie ramkę o danej liczbie bajtów   
**b) należy usunąć parametry ze stosu w procedurze lub po wyjściu z procedury w programie głównym**c) parametry pobiera sie za pomocą POP   
d) na stosie znajdują sie tylko ramki parametrów

**31.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”**

**32. W procedurze A zagnieżdżona jest procedura B, w procedurze B zagnieżdżona jest procedura C. Procedury utworzono dyrektywami PROC i ENDP. Jeżeli etykieta jest lokalna w procedurze B, to:**

* 1. Jest dostępne z procedury C i innych zagnieżdżonych w B.
  2. Jest niedostępne z procedury A, a dostępne z innych procedur zagnieżdżonych w procedurze A.
  3. Jest dostępna z procedury A, a niedostępna z innych procedur zagnieżdżonych w procedurze A.
  4. Jest dostępna tylko w procedurze B.

**33. Jeżeli procesor posiada rozkazy trójadresowe oznacza to, że:**

a) kod maszynowy analogicznego programu jest dłuższy, niż w przypadku użycia rozkazów o mniejszej ilości argumentów

b) taki rozkaz zawiera 3 pola przeznaczone wyłącznie na adresy argumentów

**c)kody maszynowe takich instrukcji są dłuższe niż w przypadku użycia rozkazów o mniejszej ilości argumentów**

d) żadna z powyższych

**34. Które ze zdań odnoszących się do dynamicznego tworzenia zmiennych lokalnych** **procedury nie jest prawdziwe:**

* 1. Tę 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. Dynamiczne zmienne lokalne nie mają lokalizacji w pamięci, dlatego nie można się do nich odwołać z procedury nadrzędnej.

**35. 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**

**36. Format pojedynczy liczby zmiennoprzecinkowej wg ISO:**

**a) zawiera wykładnik z bitem znaku**

b) zawiera 52 bitową mantysę

c) zawiera 11 bitowy wykładnik

d) zawiera mantysę w systemie znak-moduł

**37. Podaj jaki jest wynik przesunięcia w prawo liczby 11101110 w systemie U2:**

a) 01110111

b) 11011100

**c) 11110111**

d) 11011101

**38. Co powoduje zamieszczenie na początku procedury takich instrukcji:**

push EBP

mov EBP, ESP

**a) rezerwacja ramki dla procedury (albo jej zmiennych, nie pamiętam dokładnie)**

b) rezerwacja dla zmiennych lokalnych

c) przekazanie parametrów do procedury

d) dostęp do zmiennych na stosie

**39. Wskaż zdanie prawdziwe:**

a) program konsolidujący tłumaczy program składający się z kilku plików źródłowych

b) asembler dokonuje relokacji programu ta aby mógł być załadowany w dowolne miejsce

**c) loader umożliwia załadowanie programu we wskazane miejsce**

d) makroasembler tłumacząc program asm na ml dokonuje konsolidacji programu głównego   
z modułami zawierającymi definicje makrokonstrukcji

**40. Które ze zdań odnoszących sie do dynamicznej konsolidacji jest prawdziwe:**

**b) w celu wywołania procedury konsolidowanej dynamicznie sterowanie zostaje przekazane najpierw do systemu operacyjnego który inicjalizuje procedurę**

**41. Jeżeli plik wynikowy asemblera przetwarzającego program ASM ma postać konsolidowaną, to:**

a) zawiera listę pozycji relokowalnych

b) nie zawiera listy pozycji relokowalnych

**c) musi zawierać tablicę symboli lokalnych**

d) zawiera listę pozycji relokowalnych i tabelę symboli lokalnych

**42. Kolejność czynności w asemblacji:**

**(a) określenie liczby bajtów zajmowanych przez każdą instrukcję**

**(b) przypisanie instrukcjom kodów**

**(c) analiza leksykalna**

**(d) analiza składniowa**

**b) CDAB**

**43. W rejestrze warunkowym mamy bity N i Z, takie że:**

**N=1 <=> <0**

**Z=1 <=> =0**

**jaki warunek musi być spełniony aby wartość była <= 0 ?**

a) N i Z =1

b) N lub Z =0

c) N i Z =0

**d) N lub Z =1**

**44. Który z podanych zapisów uzupełnienia do 2 przedstawia liczbę -5 zapisana na 8 bitach:**

a) 11111111

b) 10000001

**c) 11111011**

d) 10001111

**45. Zapisz 0101 0111 0110 w bcd:**

**a) 576**

b) 675

c) 956

d) \*

**46. Podaj wynik w systemie poligraficznym liczb (mantysa jako znak-moduł, wynik z przesunięciem 127):**

0000 0001 @ 0111 1111 + 0000 1111 @ 1000 0000

a) 0001 0000 @ 0111 1111

b) 0000 1111 @ 1000 0000

**c) 0000 1111 @ 0111 1111**

d) 0001 0000 @ 1000 0000

**47. Jaki będzie wynik dodawania dwóch liczb w U2: 00101000+00101011=?**

**c)**

**48. Jaki będzie wynik różnicy algebraicznej 0111111 - 10000001**

**a)**

**49. Który zapis heksadecymalny odpowiada liczbie binarnej 1100 1011 1110 1011**

**a) CBEB**

b) DCFA

c) BADB

d) DCFB

**50. Kiedy nastąpi przepełnienie, przy dodawaniu liczb 8 bitowych w systemie znak-dopełnienie do 2. (C(n,m) oznacza przeniesienie z bitu n na m)**

**a) C(7,8) plus\_w\_kółeczku C(6,7)**

b) C(8,9) plus\_w\_kółeczku C(7,8)

c) C(7,8) ^ C(6,7)

d) C(8,9) ^ C(7,8)

**51. Czy za pomocą dyrektywy ORG (Orgin) można utworzyć blok danych?**a) Nie można   
**b) Tak, można**c) Nie istnieje dyrektywa tego typu   
d) Tak, pod warunkiem ze blok składa się tylko z dwóch pól.

**52. Wskaz zdanie prawdziwe dotyczące dyrektywy „ENDP”**a) Musi wystąpić na końcu każdego rozgałęzienia procedury   
b) Zastępuje instrukcje RET   
c) Umożliwia asemblerowi określenie zakresu lokalności wewnętrznych zmiennych procedury   
**d) Stosowanie jednocześnie z dyrektywa PROC umożliwia asemblerowi określenie zakresu lokalności wewnętrznych zmiennych procedury**

**53. Jakie korzyści można odnieść z utworzenia bloków programu i rozmieszczenia ich w pamięci w innej konfiguracji niż konfiguracja programu źródłowego?**a) Nie istnieją żadne korzyści   
b) Możliwość zastosowania dłuższych skoków   
c) Możliwość nie umieszczania w pliku wynikowym bloku z danymi niezainicjalizowanymi   
**d) zmniejszenie ilości pamięci zajmowanej przez program**

**54. Jeżeli w odniesieniu do parametru przekazywanego pomiędzy programem a procedura „procedura nie modyfikuje zmiennej odpowiadającej temu parametrowi”, to taki parametr określamy jako**a) wejściowo - wyjściowy   
**b) wejściowy**   
c) wyjściowy   
d) stała programu

**55. Relokacji wymagają instrukcje:**a) wszystkie   
b)z adresowaniem natychmiastowym   
c) z adresowaniem względnym   
**d) wszystkie z wyjątkiem okresowych w pkt B i C powyżej**  
**56. Relokacja kodu programu podczas ładowania programu może polegać na**a) Zastąpieniu dotychczasowych rozkazów nowymi rozkazami wynikającymi z bieżącego rozmieszczenia programu w pamięci   
**b) Dodaniu adresu początkowego położenia programu do adresów odwołań w tych rozkazach, które zostaną rozmieszczone w innych lokalizacjach niż lokalizacje ustalone podczas asemblacji programu**c) Dodaniu adresu początkowego położenia programu do adresów odwołań w tych rozkazach które odwołują się do lokalizacji zmienionych w stosunku do położenia ustalonego podczas asemblacji programu   
d) Żadne z powyższych 

**57. Które ze zdań, odnoszących się do dynamicznego tworzenia zmiennych lokalnych procedury, nie jest prawdziwe:**a) Do tej samej procedury, z dynamicznie utworzonymi zmiennymi lokalnymi, mogą odwoływać się różne równolegle wykonywane programy   
b)aby dynamicznie utworzyć zmienne procedury należy zarezerwować odpowiednia ilość miejsca na stosie i wywołać procedurę   
c) procedurę z tak utworzonymi zmiennymi lokalnymi można wywoływać rekurencyjnie   
d) dynamiczne zmienne lokalny nie maja lokalizacji w pamięci dlatego nie można się do nich odwoływać z procedury nadrzędnej.   
  
**58. Implementacja pętli WHILE polega na:**a) skok warunkowy poza pętle; na końcu pętli sprawdzanie warunku i skok bezwarunkowy na początek   
**b) sprawdzenie warunku, skok warunkowy poza pętlę: na końcu**pętli skok bezwarunkowy na początek   
c) sprawdzanie warunku, skok bezwarunkowy poza pętlę: na końcu pętli skok warunkowy na początek   
d) żadne z powyższych

**59. 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

**60. Wskaż zdanie prawdziwe nt. makroinstrukcji (MI):   
a) MI jest nazwanym ciągiem instrukcji, który zostanie wstawiony przez asembler do programu źródłowego, ilekroć**asembler napotka nazwę tej MI   
b) Użycie Mi daje identyczny skutek jak użycie procedury   
c) Użycie Mi jest mniej wygodne niż użycie procedury, gdyż   
w przypadku MI nie można przekazywać parametrów   
d) MI służą do utworzenia modułów

**61. 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
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**

**62. Jaką postać powinny mieć moduły programu aby mogły zostać podane konsolidacji statycznej?**   
a) Programu źródłowego (plik tekstowy)   
**b) Relokowalnego pliku wynikowego z tablicą symboli importowanych i eksportowanych**c) Pliku (wynikowego) ładowalnego   
d) Relokowalnego pliku wynikowego z tablicą symboli importowanych

**63. 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   
a) X-Y-Z-W-Q   
**b) Q-Y-W-X-Q**c) Q-X-Q   
d) Q-Z-W-Y-X-Y-Z-Q

**64. 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-**

**65. Wskaż zdanie nieprawdziwe dotyczące deklarowania   
procedury dyrektywną „PROC”   
a) Użycie „PROC” przy etykietce wejściowej procedury służy wskazaniu, od którego miejsca w programie rozciąga się lokalność zmiennych zadeklarowanych w procedurze.**b) Użycie „PROC” z następującą po niej lista parametrów umożliwia zadeklarowanie zmiennych wewnętrznych ze wskazaniem ich typów i utworzenie odpowiedniej ramki na stosie.   
c) Użycie „PROC” z następującą po niej listą parametrów umożliwia zadeklarowanie parametrów przekazanych między programem wywołującym a procedurą i określenie ich typów   
d) Dyrektywna „PROC” powinna być poprzednio etykietką (nazwą) procedury

**66. Do czego służy dyrektywa ASSUME w MASM 86 ?**a) Do załadowania rejestru segmentowego   
b) Do sprawdzania stanu rejestru segmentowego   
**c) Do poinformowania o zawartościach rejestru segmentowego**d) Do usunięcia zawartości rejestru segmentowego

**67. Jeżeli w programie znajduje się dyrektywa INVOKE z nazwą procedury i następująca po niej ilością parametrów, to:**   
**A. Makroasembler umieści kodzie programu instrukcje zapisania parametrów na stos i wywołanie procedury**B. Makroasembler umieści w kodzie programu instrukcje wywołania procedury, a w dalszej kolejności instrukcję tworzącą ramkę na stosie dla parametrów lokalnych   
C. makroasembler nie utworzy żadnych instrukcji poza instrukcja „wywołania procedury”, natomiast sprawdzi zgodność typów parametrów z zadeklarowanymi dyrektywa prototypu procedury („PROTO”) i w przypadku niezgodności zasygnalizuje błąd   
D. Żadne powyższe..

**68. 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**

**68. 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

**69. 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.

**70. 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

**71. 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

**72. 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**

**73. 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**

**74. 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

**75. 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:**

**76.** **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**

**77. 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.

**78. Wskaż zdanie NIE prawdziwe 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

**79. 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

**80. 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

**81. 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**