J.NAWROCKI, M. ANTCZAK, H. ĆWIEK, W. FROHMBERG, A. HOFFA, M. KIERZYNKA, S. WĄSIK

ASSEMBLER

ŚRODOWISKO URUCHOMIENIOWE

1. Pobrać plik ***masm.zip*** (*Macro Assembler 6.15* & *Segmented Executable Linker 5.60)***.**
2. Rozpakować powyższe archiwum w katalogu roboczym (na zajęciach najprawdopodobniej C:\Temp).
3. Ważne pliki wykorzystywane podczas kompilacji:
   - *ml.exe, LINK.EXE* – wykonywalne pliki reprezentujące odpowiednio *Macro Assembler 6.15* i *Segmented Executable Linker 5.60*; są one wykorzystywane w procesie kompilacji plików źródłowych programów napisanych w języku asemblera,
   - *clean.bat* – służy do usuwania następujących plików: **.bak*, *prog.exe, prog.obj, prog.map, prog.lst, prog.com* z katalogu, w którym się znajduje,
   - *prog.asm* – kod źródłowy programu zapisany w języku asemblera (*blank.asm* – szkielet programu napisanego w języku asembler, który wystarczy uzupełnić odpowiednimi instrukcjami w celu reprezentowania przez niego określonej funkcjonalności; zawiera kod, który jest niezbędny i wymagany przez wszystkie programy pisane na ćwiczeniach),
   - *make.bat* – zawiera składnię poleceń wykorzystywanych w celu kompilacji pliku zawierającego kod źródłowy programu napisanego w języku asemblera (np.: make.bat prog, w celu kompilacji pliku *prog.asm*).

     ```
     ml /Fe'kod_wynikowy' /Fl'listing_kompilacji' /Fm'mapa_kompilacji'
     /Fo'kod_przejściowy' 'kod_zrodlowy'
     link 'kod_przejściowy',,,,,
     ```

     np.:

     ```
     ml /Feprog.exe /Flprog.lst /Fmprog.map /Foprog.obj prog.asm
     link prog.obj,,,,,
     ```

ZADANIA

ZAD. 1(*). Skompiluj i wykonaj programy prezentowane na wykładzie.

ZAD. 2. Zaznacz rzeczywiste rejestry procesora x86: a) AX, b) FX, c) BX, d) GX, e) CX

ZAD. 3. Zaznacz poprawne odpowiedzi. Instrukcja `add p, z`:

a) pozwala wykonywać operację dodawania na liczbach znajdujących się w odpowiednich rejestrach,
b) po wykonaniu operacji wynik znajduje się w rejestrze p,
c) po wykonaniu operacji wynik znajduje się w rejestrze z,
d) pozwala wykonywać operację mnożenia na liczbach znajdujących się w odpowiednich rejestrach.

(*) gwiazdką oznaczone są zadania, które nie są realizowane na ćwiczeniach i są przeznaczone do wykonania jako zadania domowe.

ZAD. 4. Jaka instrukcja kończy działanie programu napisanego w języku asemblera?

ZAD. 5. Zaznacz poprawne odpowiedzi:

a) narzędzia wykorzystywane w procesie kompilacji to tylko kompilator,
b) narzędzia wykorzystywane w procesie kompilacji to tylko linker,
c) narzędzia wykorzystywane w procesie kompilacji to kompilator i linker,
d) proces kompilacji przebiega w następujący sposób: program źródłowy jest linkowany, a następnie kompilowany,
e) proces kompilacji przebiega w następujący sposób: program źródłowy jest kompilowany, a następnie linkowany,
f) kompilator generuje kod wynikowy,
g) linker generuje kod wynikowy,
h) kompilator generuje kod przejściowy,
i) linker generuje kod przejściowy.

ZAD. 6. Zaznacz poprawne odpowiedzi:

a) narzędzie do testowania kodu wynikowego to masm,
b) narzędzie do testowania kodu wynikowego to debug,
c) narzędzie do testowania kodu wynikowego to link,
d) liczby w rejestrach procesora są przechowywane w postaci binarnej,
e) liczby w rejestrach procesora są przechowywane w postaci dziesiętnej,
f) liczby w rejestrach procesora są przechowywane w postaci heksadecymalnej.

ZAD. 7. Załóżmy, że kompilowany jest plik źródłowy zawierający program napisany w języku asemblera. Uzupełnij brakujące informacje:

a) w wyniku procesu kompilacji można uzyskać następujące pliki:
   - kod przejściowy przechowywany w pliku o rozszerzeniu ……
   - kod wynikowy przechowywany w pliku o rozszerzeniu ……
   - listing kompilacji przechowywany w pliku o rozszerzeniu ……
   - mapę kompilacji przechowywaną w pliku o rozszerzeniu ……
b) w celu uzyskania informacji dotyczących możliwych opcji narzędzia *ml*, które mogą zostać wykorzystane podczas kompilacji kodu źródłowego napisanego w języku asemblera należy wykonać następujące polecenie: `ml` ……
c) w celu uzyskania informacji dotyczących możliwych opcji narzędzia *debug*, które mogą zostać wykorzystane podczas testowania poprawności kodu wynikowego uzyskanego w wyniku kompilacji należy wykonać następujące polecenie: `-`……
d) ……. – przykładowa polecenie uruchamia program *debug* w celu testowania poprawności kodu źródłowego *prog.exe*
e) jakie będą rezultaty wykonania poniższych operacji:
   -
     ```
     -rax
     AX 0000
     :1
     ```
     ……
   -
     ```
     -rax
     AX 0003
     :
     ```
     ……

(*) gwiazdką oznaczone są zadania, które nie są realizowane na ćwiczeniach i są przeznaczone do wykonania jako zadania domowe.

- `-g`
  .....
- `-q`
  .....

ZAD. 8. Oblicz metodą **bezpośredniego dodawania** wartości następujących wyrażeń (liczby są podane w systemie **heksadecymalnym**, czyli szesnastkowym; wynik ma być również heksadecymalny)(*4-6):

524	735	345	2A8	E45	CCE
+ 214	+ 879	+ CBA	+ 34C	+ 28B	+ ABA

ZAD. 9. Oblicz metodą **bezpośredniego odejmowania** wartości następujących wyrażeń (liczby są podane w systemie **heksadecymalnym**; wynik ma być również heksadecymalny) (*4-6):

548	EAB	189	437	1F2	E49
- 211	- 341	- 67	- 78	- BC	- AC8

ZAD. 10. Do czego służy kod uzupełnień do dwóch (U2) oraz jaka jest jego charakterystyczna cecha?

ZAD. 11. Podaj **zakres** liczb całkowitych, jakie można zakodować za pomocą **5 bitów** metodą uzupełnienia do 2.

ZAD. 12. Podaj wartość bezwzględną (w systemie heksadecymalnym) następujących liczb całkowitych zakodowanych na 16 bitach metodą uzupełnienia do 2 (*4-6):

FFFE FFF7 FFD3 FEF9 F30E 88FF

ZAD. 13. Zaznacz poprawne odpowiedzi. Zmiana znaku podczas tworzenia liczby ujemnej, przy wykorzystaniu jej dodatniego odpowiednika, jest wykonywana poprzez:

a) zanegowanie bitów i dodanie 1,
b) dodanie jedynki i zanegowanie bitów.

ZAD. 14. Przedstaw podane liczby ujemne w uzupełnieniu do 2 na szesnastu bitach i w kodzie heksadecymalnym (*4-6):

-4 -B -E -23 -C4 -A8

ZAD. 15. Zaznacz poprawne odpowiedzi. Instrukcja `sub p, z`:

a) pozwala wykonywać operację dodawania na liczbach znajdujących się w odpowiednich rejestrach,
b) wynik po wykonaniu operacji znajduje się w rejestrze p,
c) wynik po wykonaniu operacji znajduje się w rejestrze z,
d) pozwala wykonywać operację odejmowania na liczbach znajdujących się w odpowiednich rejestrach.

ZAD. 16. Zaznacz poprawne odpowiedzi. Po wykonaniu instrukcji `neg c`:

a) w rejestrze c znajduje się wartość taka sama jak przed wykonaniem operacji,
b) w rejestrze c znajduje się wartość powstała poprzez zanegowanie postaci binarnej liczby c.

(*) gwiazdką oznaczone są zadania, które nie są realizowane na ćwiczeniach i są przeznaczone do wykonania jako zadania domowe.

ZAD. 17. Procesor wykonuje rozkazy zapisane w kodzie wynikowym programu w następujący sposób:

……,

licznik rozkazów to ……..., który wskazuje na …….

ZAD. 18. Skoki warunkowe są realizowane za pomocą operacji porównania ……., i operacji skoku. Podaj instrukcje asemblerowe odpowiadające następującym operacjom skoku:

a) **j**ump if **e**qual - ……
b) **j**ump if **n**ot **l**ess - ……
c) **j**ump if **g**reater - ……
d) **j**ump if **n**ot **g**reater - ……
e) **j**ump if **l**ess - ……

ZAD. 19. Zapisz w języku asemblera pętle **while** oraz instrukcję warunkową **if-else**.

ZAD. 20. Napisz program przesyłający do rejestru AX **najmniejszą z liczb** znajdujących się w rejestrach BX, CX i DX. Oto przykładowe przypadki testowe:

Dane wejściowe (hex)			Wynik
BX	CX	DX	AX
1	2	3	1
5	FF	FFFF	FFFF

ZAD. 21. Napisz program przesyłający do rejestru AX **resztę z dzielenia** liczby naturalnej znajdującej się w rejestrze BX przez liczbę dodatnią znajdującą się w rejestrze CX. Zastosuj **metodę wielokrotnego odejmowania**. Oto przykładowe przypadki testowe przedstawione w systemie dziesiętnym:

Dane wejściowe		Wynik
BX	CX	AX
6	5	1
16	5	2
4	5	4
5	5	0
F	4	3

ZAD. 22. Napisz program obliczający wartość funkcji n!. Przyjmij, że *n* jest w rejestrze BX, a wynik ma być w rejestrze AX. Do mnożenia wykorzystaj instrukcję

```
imul s
```

która przesyła do pary rejestrów DX:AX wynik mnożenia liczby znajdującej się w rejestrze AX przez liczbę znajdującą się w rejestrze (lub innym miejscu) określonym przez operand *s*. Jeżeli wynik mieści się na 16 bitach i jest **nieujemny**, to wartość rejestru DX jest równa 0 (jeśli wynik mieści się na 16 bitach i jest **ujemny**, to wartością rejestru jest FFFF). Na przykład instrukcja

```
imul bx
```

(*) gwiazdką oznaczone są zadania, które nie są realizowane na ćwiczeniach i są przeznaczone do wykonania jako zadania domowe.

może być opisana następującą instrukcją w języku C:

```
ax = ax * bx;
```

Oto przykładowe dane testowe:

Wejście: n	Wynik: n!
BX	AX
0	1
1	1
2	2
3	6
4	18

Jaka jest największa wartość n, dla której n! może być poprawnie przedstawiona w uzupełnieniu do 2 na 16 bitach?

Jak należałoby zmodyfikować ten program, gdyby osoby korzystające z tego programu doszły do wniosku, że będzie im wygodniej, gdy n będzie podane w rejestrze ax, a wynik będzie w rejestrze bx?

ZAD. 23. Napisz program obliczania **liczby cyfr** dziesiętnych podanej liczby *n*. Skorzystaj z instrukcji

```
idiv s
```

która dzieli zawartość pary rejestrów DX:AX przez zawartość *s* i iloraz przesyła do rejestru AX, zaś resztę do rejestru DX. W przypadku, gdy dzielna jest 16-bitowa należy pamiętać o wyzerowaniu rejestru DX. Na przykład instrukcja języka C

```
ax = ax / bx;
```

mogłaby być przetłumaczona w następujący sposób:

```
mov dx, 0
idiv bx
```

Oto przykładowe dane testowe:

Wejście: n	Wynik:
DX	CX
D	2

ZAD. 24(*). Napisz program obliczania **wartości a^n**, gdzie *a* i *n* są liczbami naturalnymi (a>0, *n* >= *0*). Skorzystaj z instrukcji **imul** opisanej w zadaniu 22.

Oto przykładowe dane testowe:

Wejście: n	Wynik: a^n	Wejście: a
BX	AX	CX
3	8	2

ZAD. 25(*). Napisz program obliczania **sumy cyfr** dziesiętnych podanej liczby *n*. Skorzystaj z instrukcji **idiv** opisanej w zadaniu 23.

(*) gwiazdką oznaczone są zadania, które nie są realizowane na ćwiczeniach i są przeznaczone do wykonania jako zadania domowe.

Oto przykładowe dane testowe:

Wejście: n	Wynik:
DX	**CX**
D	4

(*) gwiazdką oznaczone są zadania, które nie są realizowane na ćwiczeniach i są przeznaczone do wykonania jako zadania domowe.