

問 1 picoc09 のコード生成に関する次の(1)~(2)の問いに答えなさい。

- (1) 引数として与えられた数値を表示する picoc09 の組み関数 write に対する機械語コードを下に示す。空欄に入る適切な命令を答えなさい。

#組み関数 write(int n)

write:

enter

①	3
②	
③	
④	

- (2) picoc09 の WHILE 文に対するコード生成について、下のプログラムの空欄に入る適切な語句を(a)~(j)から選びなさい。ただし WHILE 文の構文は次のとおりである。

<WHILE 文> → while (<式>) <文>

void parse_while_statement(void)

```
{
    char    l1[LABEL_LEN], l2[LABEL_LEN];
    format_label(label_counter, l1);
    label_counter++;
    format_label(label_counter, l2);
    label_counter++;
    codegen_put_label( ① );
    nextsym = scanner_get_next_sym();
    if (nextsym.sym != ② ) {
        ERROR("Parser error");
    }
}
```

nextsym = scanner_get_next_sym();

③;

if (nextsym.sym != SYM_RPAREN) {
 ERROR("Parser error");

}

codegen_put_code_str(④, l2);

nextsym = scanner_get_next_sym();

⑤;

codegen_put_code_str(⑥, ⑦);

codegen_put_label(⑧);

}

- (a) "jf" (b) "jp" (c) "jt" (d) l1 (e) l2 (f) parse_expression()
(g) parse_statement() (h) SYM_LPAREN (i) SYM_RPAREN (j) SYM_WHILE

- (3) 次のプログラムに対して生成される picoc09 の機械語コードを示しなさい。ただし、変数 a の変位は A で表すものとする。

if(a) a = 0;

問2 構文解析に関する下の(1)~(3)に答えなさい。

- (1) ①~④を、解析能力が低いものから高いものへ順番に並べなさい。
 (2) 上向き構文解析手法に属するものの番号をすべて示しなさい。
 (3) LR 構文解析手法の中で、解析能力と構文解析表サイズのバランスの点ですぐれており、実際の言語処理系でよく用いられるものを番号で示しなさい。

① LALR 構文解析法 ② LL 構文解析法 ③ SLR 構文解析法 ④ 正準 LR 構文解析法

問3 下の①~⑦の規則で表される文法に関して次の(1)~(10)の問いに答えなさい。

- (1) 規則②のように左辺の構文変数 (E) が右辺の式の先頭に現れることをなんと呼ぶか。
 (2) 下の SLR 構文解析表において、括弧なしの数の欄に対応する動作はなんと呼ばれるか。
 (3) 下の SLR 構文解析表において、括弧つきの数の欄に対応する動作はなんと呼ばれるか。
 (4) 下の SLR 構文解析表において、括弧内の数は何を指しているか。
 (5) 項 (たとえば「 $E' \rightarrow \cdot E$ 」) における「 \cdot 」をなんと呼ぶか。
 (6) 規則④について、項をすべて示しなさい。
 (7) 項「 $E' \rightarrow \cdot E$ 」の閉包を求めなさい。
 (8) SLR 構文解析において、項「 $A \rightarrow x \cdot$ 」を含む閉包を解析しているときに、 x を A に還元するかどうかの判断基準を、FOLLOW 集合を用いて説明しなさい。
 (9) 下の SLR 構文解析表を用いて、入力「 $(id + id) * id$ 」に対する構文解析の動作を示しなさい。
 (10) 入力「 $(id + id) * id$ 」に対する解析終了時のスタックと残り入力を示しなさい
 ※(9), (10)は解答用紙の表の空欄に記入

文法 ① $E' \rightarrow E$ ② $E \rightarrow E + T$ ③ $E \rightarrow T$ ④ $T \rightarrow T * F$
 ⑤ $T \rightarrow F$ ⑥ $F \rightarrow id$ ⑦ $F \rightarrow (E)$

SLR 構文解析表

状態	状態遷移								
	id	+	*	()	\$	E	T	F
0	5			4			1	2	3
1		6				終了			
2		(3)	7		(3)	(3)			
3		(5)	(5)		(5)	(5)			
4	5			4			8	2	3
5		(6)	(6)		(6)	(6)			
6	5			4				9	3
7	5			4					10
8		6			11				
9		(2)	7		(2)	(2)			
10		(4)	(4)		(4)	(4)			
11		(7)	(7)		(7)	(7)			

アセンブラ表記	意味	動作
pushl <i>n</i>	ローカル変数をプッシュ	push(*(fp + n));
storel <i>n</i>	ローカル変数への格納	*(fp + n) = *sp;
storet <i>n</i>	一時領域への格納	*(sp + n) = *sp;
pushi <i>d</i>	定数をプッシュする	push(d);
call <i>label</i>	関数を呼び出す	push(pc); pc = label;
ret	関数から復帰する	pc = pop();
enter	スタックフレームを生成	push(fp); fp = sp;
leave	スタックフレームを解放	sp = fp; fp = pop();
mvsp <i>n</i>	SPを移動する	sp = sp + n;
jp <i>label</i>	無条件に分岐する	pc = label;
jt <i>label</i>	真なら分岐する	if (pop() != 0) pc = label;
jf <i>label</i>	偽なら分岐する	if (pop() == 0) pc = label;
add (sub, mul, div, mod)	加算を行う (減 / 乗 / 除 / 剰余)	t1 = pop(); t2 = pop(); push(t2 + t1);
le (eq, ne, gt, ge, lt)	「<=」のとき真 (==, !=, >, >=, <)	t1 = pop(); t2 = pop(); if (t2 <= t1) push(1); else push(0);
rd	値の読み込み	scanf("%d", &t); push(t);
wr	値の出力	t = pop(); printf("%d ", t);
wrln	改行	printf("\n");
halt	プログラムの停止	

※*n*, *d*, *label* はオペランド

※pc, sp, fp はそれぞれプログラムカウンタ, スタックポインタとフレームポインタ
(仮想CPUのレジスタ)

※t, t1, t2などは説明のための便宜的な変数

※push, popの動作は次のとおりである。

```

push(x) {
    sp--;
    *sp = x;
}

pop() {
    t = *sp;
    sp++;
    return t;
}

```

学籍番号 J

問 1

(1) ① pushl ② wr ③ leave ④ ret

(2) ① (d) ② (h) ③ (f) ④ (b) ⑤ (g) ⑥ (a) ⑦ (d) ⑧ (e)

問 2 (1) (解析能力の低い順に) ② ③ ① ④ (2) ① ③ ④ (3) ①

問 3 (1) 左再帰性 (2) シフト (3) 還元

(4) 文法の番号 (5) 黒ボク注視点

(6) $T \rightarrow \cdot T^* F, T \rightarrow T \cdot^* F, T \rightarrow T^* \cdot F, T \rightarrow T^* F.$

(7) $\{E' \rightarrow \cdot E, E \rightarrow E + T, E \rightarrow T^* F, F \rightarrow id, F \rightarrow \cdot (E)\}$

(8) 字句を先読みし、その字句が FOLLOW(A) に含まれていれば還元できる。

(9) (10)

	スタック	残り入力
(1)	0	(id + id) * id \$
(2)	0 (4	id + id) * id \$
(3)	0 (4 id 5	+ id) * id \$
(4)	0 (4	F + id) * id \$
(4')	0 (4 F 3	+ id) * id \$
(5)	0 (4	T + id) * id \$
(5')	0 (4 T 2	+ id) * id \$
(6)	0 (4	E + id) * id \$
(6')	0 (4 E 8	+ id) * id \$
(7)	0 (4 E 8 + 6	id) * id \$
(8)	0 (4 E 8 + 6 id 5) * id \$
(9)	0 (4 E 8 + 6	F) * id \$
(9')	0 (4 E 8 + 6 F 3) * id \$
終了時	0 E 1	\$

0 (4 E 8 + 6
0 (4 E 8 + 6 T 9
0 (4
0 (4 E 8
0 (4 E 8) 11
0
0 F 3
0
0 T 2
0 T 2 * 7
0 T 2 * 7 id 5
0 T 2 * 7
0 T 2 * 7 F 10
0
0 T 2

T) * id \$
) * id \$
E) * id \$
) * id \$
* id \$
F * id \$
* id \$
T * id \$
* id \$
id
F
T
F